TRAINS

Knowledge Representation in the TRAINS-93 Conversation System

D.R. Traum, L.K. Schubert, M. Poesio, N.G. Martin,
M.N. Light, C.H. Hwang, P.A. Heeman, G.F. Ferguson, and J.F. Allen

TRAINS Technical Note 96-4 and Technical Report 633
August 1996

UNIVERSITY OF
ROCHESTER
COMPUTER SCIENCE

Knowledge Representation in the TRAINS-93 Conversation System*

David R. Traum[1] Lenhart K. Schubert[2] Massimo Poesio[3]
Nathaniel G. Martin[2] Marc Light[4] Chung Hee Hwang[2] Peter Heeman[2]
George Ferguson[2] James F. Allen[2]

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 633 and TRAINS Technical Note 96-4

Abstract

We describe the goals, architecture, and functioning of the TRAINS-93 system, with emphasis on the representational issues involved in putting together a complex language processing and reasoning agent. The system is intended as an experimental prototype of an intelligent, conversationally proficient planning advisor in a dynamic domain of cargo trains and factories. For this team effort, our strategy at the outset was to let the designers of the various language processing, discourse processing, plan reasoning, execution and monitoring modules choose whatever representations seemed best suited for their tasks, but with the constraint that all should strive for principled, general approaches.

Disparities between modules were bridged by careful design of the interfaces, based on regular in-depth discussion of issues encountered by the participants. Because of the goal of generality and principled representation, the multiple representations ended up with a good deal in common (for instance, the use of explicit event variables and the ability to refer to complex abstract objects such as plans); and future unifications seem quite possible. We explain some of the goals and particulars of the KRs used, evaluate the extent to which they served their purposes, and point out some of the tensions between representations that needed to be resolved. On the whole, we found that using very expressive representations minimized the tensions, since it is easier to extract what one needs from an elaborate representation retaining all semantic nuances, than to make up for lost information.

The University of Rochester Computer Science Department supported this work.

[*] To appear in International Journal of Expert Systems Vol. 9, No. 1.

[1] TECFA, FPSE, Université de Genève, 9 Rte de Drize, 1227 Carouge, Switzerland.
Email: David.Traum@tecfa.unige.ch Telephone Number: +41 22 705 9696 Fax: +41 22 342 8924

[2] Computer Science Department, University of Rochester, Rochester, NY 14607, USA.

[3] Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh, EH8 9LW Scotland.

[4] Universität Tübingen, SfS Computerlinguistik, Wilhelmstr. 113, D-72074 Tübingen, Germany.

REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE August 1996	3. REPORT TYPE AND DATES COVERED technical report

4. TITLE AND SUBTITLE

Knowledge Representation in the TRAINS-93 Conversation System

5. FUNDING NUMBERS

ONR/ARPA N00014-92-J-1512

6. AUTHOR(S)

Traum, Schubert, Poesio, Martin, Light, Hwang, Heeman, Ferguson, & Allen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESSES

Computer Science Dept.
734 Computer Studies Bldg.
University of Rochester
Rochester NY 14627-0226

8. PERFORMING ORGANIZATION

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESSES(ES)

Office of Naval Research
Information Systems
Arlington VA 22217

ARPA
3701 N. Fairfax Drive
Arlington VA 22203

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

TR 633 and TRAINS TN 96-4

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution of this document is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

(see title page)

14. SUBJECT TERMS

knowledge representation; natural language understanding; dialogue systems

15. NUMBER OF PAGES

52 pages

16. PRICE CODE

free to sponsors; else $3.00

17. SECURITY CLASSIFICATION OF REPORT unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT unclassified	20. LIMITATION OF ABSTRACT UL

Key Words: Knowledge Representation, Natural Language Understanding, Dialogue Systems

1 Introduction

TRAINS is a long-term research project aimed at developing an intelligent, conversationally proficient planning assistant. The task domain for the initial phases of the project involved the use of factories and cargo trains to achieve given delivery goals, such as the delivery of some quantity of orange juice to particular cities, using orange juice factories and orange warehouses at other cities in a railroad network. We have built a prototype system at the end of each of the first 5 years of the project. Each year's effort, except for the most recent (TRAINS-95), was centered around one or two particular problem solving dialogues, obtained by slightly editing transcripts selected from a corpus of human-human spoken dialogues involving TRAINS-world problems. An overview of the TRAINS project up to 1994 appears in [Allen et al., 1995b].

In the present paper we focus on the interactions between the language understanding components and reasoning components in TRAINS-93, which allows the TRAINS modules to work together in an incremental, cooperative planning process. The nature of these interactions depends crucially on the nature of the knowledge representations used within the interacting modules. Thus much of the paper will be devoted to explaining the rationales for the various representations, the way in which inter-module communication in TRAINS-93 bridges the gaps between the representations, and the advantages as well as the remaining tensions associated with this architecture.

The heterogeneous architecture of TRAINS-93 is a consequence of the project goals and our research philosophy. Our primary goal was, and remains, to gain a deeper understanding of the relationship between NL dialogue and problem solving, how each informs and constrains the other. How, we ask, is dialogue shaped by the need to solve realistic problems involving significant amounts of world knowledge? And conversely, how is cooperative problem-solving shaped by the availability of and necessity for linguistic interaction?

In addition, the TRAINS project is intended to serve certain "research management" objectives. It provides a setting in which researchers on different facets of NLP and problem solving can confront the interface issues and work out a common, integrated, comprehensive view of problem solving dialogues, as opposed to working independently on just a few selected problems of theoretical interest. Finally, we are committed to the practical goal of building a series of working and integrated prototype systems, as a way of keeping the theoretical efforts focused and earth-bound. While these systems are not ends in themselves, they are intended as steps toward a conversationally proficient, intelligent planning assistant for dynamic task domains.

As indicated in our opening remarks, our methodology has been to undertake a series of joint efforts centered around specific, increasingly complex and realistic dialogues. By drawing these dialogues from a corpus of natural spoken dialogues between people, we ensure that our theories will apply to actual discourse, and that our systems will show increasingly "life-like" behavior. However, we would not be satisfied with *ad hoc* theories and techniques that happen to work in the TRAINS domain. We are interested in principled, general solutions that will apply to entirely different and potentially richer task domains. Given this emphasis on generality, it was important to let individual researchers bring to bear whatever representational theories and techniques seemed to them most promising for their subproblem. In this, our system differs from the SOAR project [Lehman et al., 1991] which is similar in its aims to build an integrated, multi-functional, language-using agent. The SOAR project uses a common knowledge representation and inference mechanism throughout. We decided, instead, to seek integration through careful consideration of interface and control issues, rather than insisting on a uniform representation from the outset; efforts on individual modules are kept "honest" by the need to interact with other modules in a straightforward way.

The rest of the paper is organized as follows. In the next section we present an overview of the TRAINS domain and the modules of the TRAINS-93 system. Section 3 describes the knowledge requirements of various aspects of the system. Next we present an overview of the KR languages used in the system, in Section 4, followed by a trace of the system working on a small dialogue fragment, in Section 5. We continue with a discussion of how the different languages are related and some of the obstacles to unification of the ontologies inherent in the different KR systems, and conclude with an evaluation of the state of the system and future directions.

2 The TRAINS-93 System

In this section we briefly describe the domain and architecture of the TRAINS-93 system, in order to provide a concrete foundation for the specific discussion of the knowledge representation and reasoning issues in the rest of the paper. More details on the system itself, and rationales for choosing the domain can be found in [Allen et al., 1995b].

2.1 The TRAINS Domain

The TRAINS-93 system helps a user construct and monitor plans about a railroad freight system. The user is responsible for assigning cargo to trains, scheduling cargo shipments and various simple manufacturing tasks, and for revising the original plans when unexpected situations arise during plan execution. Neither system nor user directly executes the actions in the TRAINS domain. Rather, the actions are performed by the simulated railroad engineers and factory and station supervisors. Dialogue is encouraged by giving the user and the system different responsibilities and knowledge. The user is given the goals that need to be achieved but does not have direct access to the agents in the world or complete knowledge about the actions that can be performed. The system acts as the user's assistant. It knows what actions are possible, and provides the communication link to the world. It interacts with the user about all aspects of the task, using a natural language interface. The only information that they initially share is a map of the current TRAINS scenario, as shown in Figure 1. The world consists of five cities, each of which contains a rail station that contains engines and rail cars as indicated. Some cities also contain warehouses containing goods

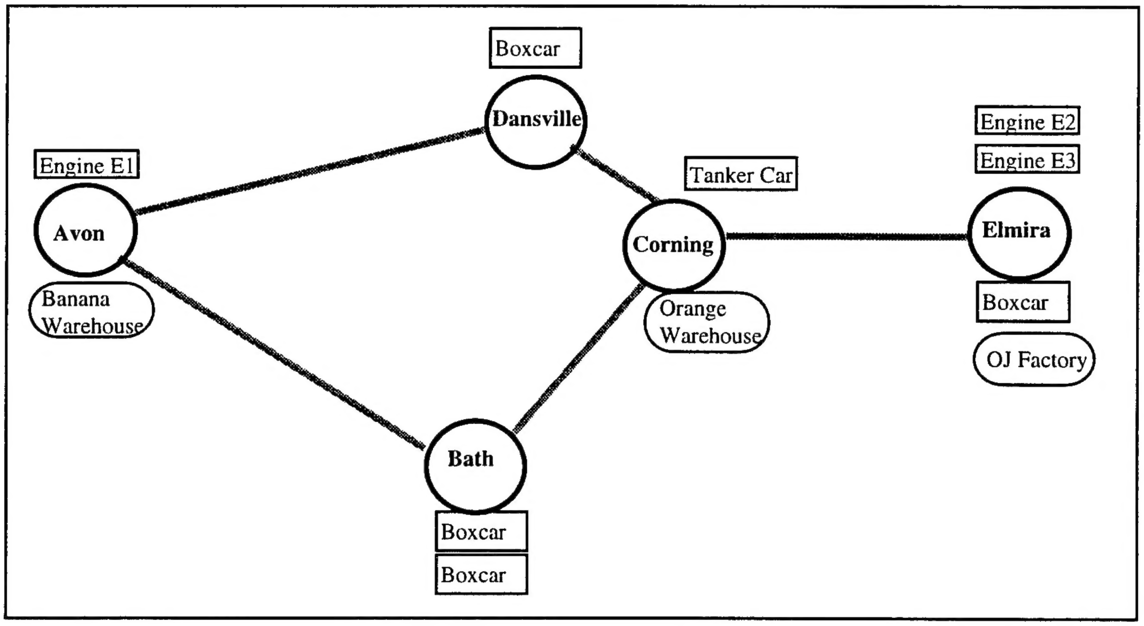


Figure 1: A TRAINS-world Map

(*e.g.*, there are bananas at Avon and oranges at Corning), and factories (*e.g.*, Elmira has an orange juice factory that converts oranges to orange juice).

Our research has been strongly motivated by studying actual human performance in these situations. We have collected over eight hours of human-human dialogue[1] in the TRAINS domain [Gross et al., 1993; Heeman and Allen, 1994b], and have used this data to gain insight into both how people collaborate to form plans and how language is actually used to construct and discuss joint plans. This type of analysis is crucial for identifying what problems actually arise in dialogue, and helps us avoid working on issues that might seem interesting in the abstract, but do not occur in practice.

Figure 2 shows the target dialogue for the TRAINS-93 implementation. The dialogue concerns the TRAINS world situation given in Figure 1, with the goal, as stated in the first utterance, of shipping a boxcar of oranges to Bath. This dialogue is a simplified version of one of the human-human dialogues from our corpus.[2] The simplifications include deletions of disfluencies, elimination of some difficult interactions (*e.g.*, elimination of a difficult repair subdialogue in turns 9 to 13), and merging of separate prosodic phrases into full sentential utterances (*e.g.*, 3.3-3.6 merges the third to sixth prosodic phrases of the third turn in the original spoken dialogue into one complete sentential utterance). The labeling scheme here thus maintains a correspondence to the `<turn>.<utterance>` numbering of the original, despite the simplifications.

This dialogue was chosen as a point of common ground for integrating and testing the

[1] In which a person fills the role of the system.

[2] d91-6.1 in [Gross et al., 1993].

1	U: We better ship a boxcar of oranges to Bath by 8 AM.
2	S : Okay.
3.3-3.6	U: So we need to get a boxcar to Corning where there are oranges.
3.7	U: There are oranges at Corning.
3.8	U: Right?
4	S : Right.
5.1	U: So we need an engine to move the boxcar.
5.2	U: Right?
6	S : Right.
7.1-7.2	U: So there's an engine at Avon.
7.3	U: Right?
8	S : Right.
9-13	U: So we should move the engine at Avon, engine E1, to Dansville to pick up the boxcar there.
14	S : Okay.
15.2-15.4	U: And move it from Dansville to Corning.
15.5-15.7	U: Load up some oranges into the boxcar.
15.8-15.10	U: And then move it on to Bath.
16	S : Okay.
17	U: How does that sound?
18.3	S : That's no problem.
19	U: Good.

Figure 2: The dialogue processed by TRAINS-93.

different components of the system together since it places most of the processing burden on the language understanding, knowledge representation, and plan-recognition components, while requiring only simplistic NL generation to achieve the desired system responses. The TRAINS-93 implementation can engage in the dialogue in Figure 2 with a user typing into the keyboard. The system interprets and evaluates the user's utterances and produces appropriate responses, while maintaining the dialogue context and building a representation of the shared plan which can be sent to a TRAINS world simulator and executed. Section 5 shows an annotated trace of some of the representations produced as the system processed turns y and 8 of the dialogue in Figure 2.

2.2 The TRAINS-93 System Architecture

Figure 3 shows the modules of the TRAINS-93 implementation, as well as the main flow of communication between modules. Also shown are the representation languages used by the modules. These languages will be discussed in section 4.

As the figure indicates, the language interpretation modules are tightly integrated with the task modules for domain planning and plan execution. Granularity of interleaving in the implementation is at the sentential utterance level. Each utterance (from either the user or the system) is processed by all the modules up to the dialogue manager (which calls the domain reasoner to disambiguate hypotheses about meaning and to update the representation of the current plan). Information from the planner can also be used as a basis for forming NL responses, *e.g.*, acceptances, rejections, or repairs of plan-based suggestions.

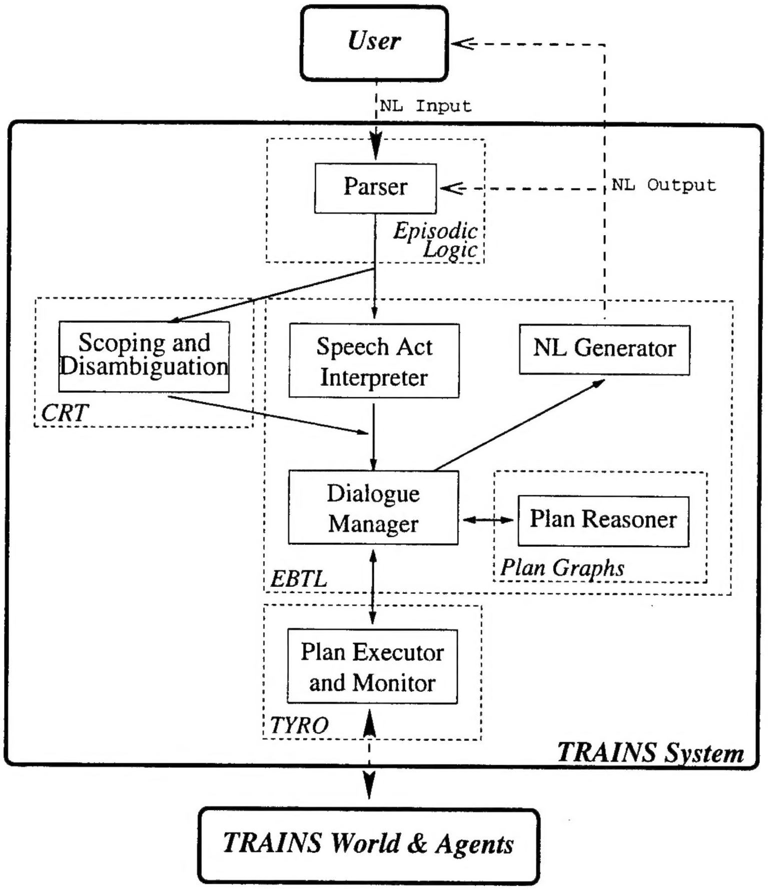


Figure 3: TRAINS-93 System Architecture

When a complete plan has been built and is believed to be shared by the conversants, this can be sent to the executor and monitor to be executed in the simulated world.

Parsing and Logical Form Computation

The first module in the interpretation process is a parser which takes an utterance as input and produces a syntactic analysis together with an underspecified logical form (ULF), allowing for scopal and referential ambiguities. We use a GPSG style grammar [Gazdar et al., 1985] that makes extensive use of a feature system including subcategorization features and several feature principles governing feature percolation. Each rule in the grammar consists of a syntactic rule coupled with a corresponding semantic rule. This allows us to build the interpretation compositionally in the sense of Montague's semantics [Montague, 1973]. The parser is a chart parser which uses probabilities gathered from the TRAINS corpus to choose which arc to expand next.

A lexicon consisting of syntactic features and semantic expressions for 700 stems was constructed by hand. In addition, the Alvey lexicon (60,000 stems) and morphological analyzer (Kimmo-based segmenter combined with a unification-based chart parser) [Ritchie et al., 1992] were integrated with the TRAINS sentence-level parser. However, the Alvey lexicon does not contain the fine-grained semantic information needed by the TRAINS system and thus was only useful for its syntactic features which were mapped straightforwardly onto the TRAINS syntactic features.

Scoping and Deindexing

The task of the Scoping and Deindexing module, SAD-93, is to 'deindex' (*i.e.*, fix the value of) context-dependent aspects of an utterance's content such as referential expressions. The input to SAD-93 is the underspecified representation produced by the parser. The output is a set of alternative hypotheses about how to resolve the ambiguities that are suggested in the given context.

SAD-93 arrives at a hypothesis by applying discourse interpretation rules with varying strength, all of which operate directly off the underspecified interpretation. The decision to integrate scope interpretation with other aspects of pragmatic interpretation reflects the thesis put forth in [Poesio, 1994] that the scoping preferences of human subjects result from the interaction of several discourse interpretation processes.

Speech Act Analysis

The speech act interpreter is responsible for determining what the speaker means by her utterance. For instance, when a speaker utters "there are oranges at Corning" (as in utterance 3.7 in the conversation presented in Figure 2), she might be saying this to *inform* the hearer of this fact, to *check* whether the hearer agrees with this, to *question* the hearer about whether this is the case, or as a *suggestion* to use the oranges in the current plan. These alternatives are not all mutually exclusive. The speech act interpreter builds a list of hypotheses about the speech act interpretations of an utterance.

Rather than working from fully scoped and deindexed forms, the speech act analyzer, like the Scoper and Deindexer, takes the ULF as its input. After both the speech act analysis and scoping and deindexing are completed, the result of scoping and deindexing is incorporated into the speech act hypotheses, which is then passed on to the dialogue manager for verification.

Dialogue Management

The dialogue manager is responsible for maintaining the flow of conversation and making sure that the conversational goals are met. For this system, the main goal is to construct and agree upon a plan with the user, and then to execute it.

The dialogue manager must keep track of the user's current understanding of the state of the dialogue, verify the hypotheses about intentions that motivate utterances of the user, adopt intentions (in the form of speech acts) to respond, send these intended speech acts to the NL generator to produce system utterances, and send commands to the domain plan reasoner and domain plan executor when appropriate. The dialogue manager is described in more detail in [Traum and Allen, 1994; Traum, 1994].

Domain Plan Reasoning

The plan reasoner provides planning and plan recognition services and performs reasoning about the state of the world. The system must explicitly represent the plan(s) under consideration, since many of the user's utterances concern contributions to the current plan. In our framework, representations of plans contain explicit statements of what the goals of the plan are and, importantly, the assumptions underlying the plan. Assumptions must be made both during plan recognition (to fill in details left unspecified by the user but necessary to connect the current utterance to the plan) and during planning (for example, persistence assumptions). Since these assumptions often drive the dialogue, we have investigated an explicit representation of plans as arguments based on assumptions [Ferguson, 1995].

The TRAINS plan reasoner supports interleaved planning and recognition (as necessary in processing dialogue), and exports a variety of functions to other modules of the system, in particular the dialogue manager. The dialogue manager uses the results of plan reasoning to disambiguate speech act interpretations, update beliefs, and generate new conversational elements (*e.g.*, an ambiguity detected by the plan reasoner could cause a clarification subdialogue to be started).

NL Generation

The NL Generator takes speech act representations produced by the dialogue manager and converts them to natural language text which is then "uttered" as output to the user (and fed back through the language analysis modules, as described in Section 5). As NL Generation was not a research focus of the project, a very simple template matcher was used to produce utterances from the speech act forms, without regard to some of the more complex generation issues (such as the sequencing of information for presentation, the use of detailed knowledge of syntax and the correspondence between syntax and semantic representation, rhetorical style, or choice of the most appropriate referring expression, *etc.*).

Plan Execution

The plan executor implements domain plans by sending requests to individual agents (engineers, factory and warehouse attendants) in the simulated world. It selects a set of requests by reasoning about the possible consequences of the actions that agents will perform in satisfying the requests.

The plan executor also monitors the progress of the plan by communicating with these agents as they perform their tasks, gathering information so it can make better choices when executing subsequent plans. It bases its decisions on constraints on probabilities that are inferred from observations, from probabilities asserted by the programmer, and from facts inferred from its knowledge base. The principles behind the executor's operation are described in more detail in [Martin, 1993].

The plan executor can also perform two additional services for other modules. It can aid the planning process by making choices among a set of alternatives (*e.g.*, choosing which among a set of alternative paths to take between destinations), and it can monitor and report the current state of the world.

TRAINS World Simulation

The TRAINS world is a detailed simulation of action executions by agents on world objects. It is used for plan execution and monitoring (since, as far as the system is concerned, it is the real world). The intentions of the system have no direct bearing on the behavior of the world. Actions will take place regardless of the system's goals. The world implements a dynamic physical model which is not accessible by the system. The only way the system can effect change in the world is by sending directives to the agents in the world — *i.e.*, the engineers, factory managers, *etc.* If they receive the messages (some may be lost), they execute the requested actions. These actions may or may not have the consequences anticipated.

3 Knowledge Requirements of Problem-Solving Dialogues

To engage in the TRAINS task and participate in a dialogue like that shown in Figure 2, a system must reason about a number of different types of knowledge. Separate research traditions involving different subtasks have led to (sometimes very) different ontologies and representations. An obvious split is between language interpretation tasks, in which the representation language must be rich enough to represent the nuances present in natural language, and domain or plan reasoning tasks, in which a representation closer to the ontology of action in the domain is desired. There are, however, separate research traditions even within these two areas.

On the language processing side, there is often a split between more semantically oriented representations, which focus on the truth-functional nature of a sentence given a model, and more pragmatically oriented representations which focus on how the utterance in context affects the conversing agents. On the task side, there is also a tension between the representations most convenient for finding efficient plans, talking about them with another agent, and executing them in the world. In the rest of this section, we describe some of these divergent representational requirements that must be addressed within the scope of a system designed to perform the TRAINS task. In Section 4, we show how these issues are addressed by the actual KR languages, some of which are exemplified in the trace in the following section. Then in Section 6, we return to the conflicting assumptions in the different research traditions and show the connections and some of the differences between the languages and why it would have been difficult to start out with a single representation language.

3.1 Knowledge of Language

When dialogues with a natural language system are limited to some narrow task domain, the form and content of utterances tend to be equally limited. In such a setting it is often possible to achieve a semblance of understanding through *ad hoc* rules that shortcut serious syntactic and semantic analysis, going more or less directly from domain-specific clue words and patterns to domain-specific meanings and intentions.

Since the TRAINS task domain is quite narrow, such shortcut methods could certainly be employed to achieve short-term performance goals (and indeed we are interested in integrating such methods with more general ones). However, as stated at the outset, our goal in TRAINS-93 was to handle language in a way that would be defensible for *any* conversational domain, even those far richer and varied than the TRAINS domain. In other words, the approach should in principle be extensible to "full understanding", with full use of all available syntactic, semantic, and world knowledge to interpret and respond to inputs. In fact, we took as our working hypothesis that the techniques and representations we had previously developed for understanding narratives (including children's stories) should carry over into the TRAINS domain.

The requirement with respect to syntax is that the chosen syntactic framework be domain independent and (potentially) comprehensive, and in a form that facilitates the syntax-semantics mapping. Also, an important requirement is the design of a feature system that is helpful in systematically minimizing syntactic ambiguity, for instance filtering out syntactically inappropriate PP attachments (such as attachment of *to Corning* to *need* in *We need to get a boxcar of oranges to Corning*).

Semantic interpretation is decidedly the most challenging aspect of building a NL module, in terms of the representational and interface issues raised. To achieve generality and extensibility, we naturally looked to formal linguistic semantics for theoretical foundations. Thus Montague semantics, situation semantics, and discourse representation theory provided important building blocks. However, any given linguistic semantic theory (with the exception of Montagovian ones) generally employs some formalism with limited expressiveness, neglecting many semantic phenomena encountered in even the most mundane dialogues. As well, each theory tends to provide only fragmentary details about the mapping from surface form to the formal meaning representation, so that it is hard to tell whether the theory could be extended to a realistic grammar with broad syntactic and semantic coverage, with a reasonably simple transduction from surface form to the meaning representation, and with an approach to the problems of context and indexicality.

By contrast, our computational goals require that we be able to derive semantic representations for anything the user might reasonably say, within a single, comprehensive framework. Thus expressiveness and ease of transduction are crucial. The two issues are related: if we can devise a representation that closely matches the expressiveness of ordinary language, in terms of the available vocabulary of concepts and the permissible ways of combining these into complex types, then it should also be relatively simple to derive the representations of utterances. These considerations can be restated in terms of the following three general requirements for a semantic representation language, which we have striven to meet in designing the target EL representation (described below in Section 4.1):

expressive adequacy: the language should be powerful enough to allow us to represent the most common constructs and semantic nuances in naturally occurring sentences. Among other things, this includes tense operators (past and present), aspect (perfect and progressive), various kinds of adverbials, including manner adverbials (*e.g.*, "using engine E3"), purpose clauses (*e.g.*, "to pick up the boxcar"), predicate modifiers (*e.g.*, "almost finished," "very good," "looks difficult"), kinds of actions and events (*e.g.*, the kind of action "to get a boxcar", referred to in "we need to get a boxcar" – see turn 3 in Figure 2), kinds of natural objects (*e.g.*, "we need oranges"), modals (*e.g.*, "need to," "should," "had better"), and other intensional verbs including creation verbs (*e.g.*, "make OJ"). A further requirement in the initial stages of interpretation is to allow for underspecified representations, *i.e.*, logical forms that remain noncommittal about the scopes of certain operators (such as quantifiers and tense operators) and about the referents of referring expressions such as pronouns.

derivational adequacy: the language should support a simple, systematic derivation of meaning representations from English surface structures.

semantic adequacy: the meaning of the language itself should be precisely defined, *i.e.*, it should have a denotational semantics. This requirement is met by the linguistic semantic formalisms that were our starting point, but is all too easily lost sight of when one devises extensions to deal with the semantic complexities of real discourse. Yet semantic adequacy is crucial for ensuring representational coherence (in the sense that the symbols we use in the meaning representations be capable of consistently carrying the meaning we *intend* them to carry), and for supporting the desired *inferences.*

3.2 Conversation Structure

In order to carry on a coherent conversation, in addition to the representation of the semantics of sentences, the system must be able to represent the context of conversation and how that context is dynamically updated during the conversation. This context consists not just of the content expressed in the utterances, but of structured information about the interaction, as well. The relevant structure which must be maintained to understand and participate felicitously in conversation includes the following:

discourse segments: Attentional focusing structure [Grosz and Sidner, 1986] is useful for a variety of purposes in linguistic interpretation and generation, including the ability to determine the possible referents for a referring expression and the intentional relations between utterances. Discourse segmentation structure, particularly the notions of focus and accessibility, will guide how certain utterances will be interpreted. Utterances like "yes" can be seen as an acceptance or confirmation of the most recent unanswered question, if one exists in a discourse segment that is still open. Certain utterances such as "by the way", or "anyway", or "let's go back to" or "let's talk about" will signal a shift in segments, while other phenomena such as clarifications will signal changes in the structure by their information content.

intentional structure: A related structure (also described in [Grosz and Sidner, 1986]) has to do with why the speakers are engaging in the particular subdialogues - how the subdialogues relate to the overall purposes of the conversation. For task-oriented conversations, the nature of the task often has a strong bearing on the intentional structure of the dialogue about the task.

grounding: The participants must track their state of mutual understanding of what has been communicated. This involves both performance and recognition of acknowledgments, and repairs, when necessary [Clark, 1992].

turn-taking: The notion of who has the turn is important in deciding whether to wait for the other agent to speak, or whether to formulate an utterance. It will also shape the type of utterance that will be made, *e.g.*, whether to use some kind of interrupting form or not. Turn-taking is also influenced by the overall initiative (or *Control* [Walker and Whittaker, 1990]), which is often related to the intentional structure. In the TRAINS domain, the initiative is shared, with different participants holding it at different points in the conversation. In the initial phase, the user has the initiative while the task is conveyed. In the main part of the conversation – the construction of the plan – the initiative can lie with either party, though it generally remains with the user. In the final phase, verifying successful completion of the problem, the initiative belongs with the system.

rhetorical: There is also often a local structuring of utterances into standard subdialogue types. Examples include pairs like questions and answers, suggestions and acceptances, as well as summaries and lists. This kind of structure is variously called *rhetorical* [Mann and Thompson, 1987], *exchange* [Sinclair and Coulthard, 1975], *adjacency pair* [Schegloff and Sacks, 1973], or *dialogue games* [Mann, 1988]. While this type of structure is somewhat similar to the intentional level, above, they are not identical, since the same rhetorical patterns may be used for a variety of purposes, yet expectations and obligations to respond appropriately within the particular *dialogue game* remains the same, regardless of underlying intentions. The precise nature of this type of structure and its relations to other types is still fairly controversial as can be seen in by the contributions in [Rambow, 1993].

situations: Non-linguistic information relevant for conversation is structured into coherent bundles or *situations.* Some of the situations that are relevant for the TRAINS task include:

the discourse situation: the speaker and hearer together with their utterances and other actions, as well as aspects of their mental states, described in the following section.

the map situation: what the user knows about the initial conditions of the world from the TRAINS map, shown in Figure 1.

plan situations: updates of the map situation representing what would be the case in the future, if particular plans are performed.

3.3 Mental State

In order to participate fully in the conversation, the system needs to represent certain aspects of both its own and its partner's mental state. In particular, the system must represent at least the following aspects of mental state.

beliefs: In addition to representing its own beliefs about the state of the world, the system must represent its beliefs about the user's beliefs in order to interpret the user's utterances. It must also represent mutual beliefs so that it can track the grounding process.

goals: The system needs to keep track of what goals it hopes to achieve to achieve by participating in the conversation. These will lead to adoption of specific intentions and performance of actions.

plans: The system must represent different views of plans, including: plans it has developed itself via planning, proposals by itself and the user, and mutually agreed-upon (or *shared*) plans. The distinctions are important for the system to respond appropriately in order to achieve the final goal of an executable shared plan, (*e.g.*, proposing an addition to the plan, or accepting or rejecting a prior suggestion).

intentions: The intentions of the system help guide the future behavior of the system, generally constraining future planning to fit within the context of prior intention. [Bratman, 1987] describes in detail some of the ways that intentions will help guide and constrain an agents future deliberation.

obligations: Obligations are actions that an agent *should* perform, according to external norms. They are different in kind from goals or intentions, though a well-behaved agent will choose to meet its obligations, and in fact there is generally a social cost for not meeting obligations which may encourage a purely strategic agent to endeavor to meet them. In a conversational setting, an accepted offer or a promise will incur an obligation. Also a request or command by the other party will bring an obligation to perform or address the requested action. Discourse obligations are discussed further in [Traum and Allen, 1994].

While some of these attitudes might be represented only implicitly in the control structure of a dialogue system, a declarative representation will help in diagnosing and repairing problems that come up in the interaction, as well as allowing a flexible initiative strategy (*e.g.*, placing more or less importance on obligations or goals at different times in the conversation). Goals and obligations are also necessary to represent the meaning of utterances such as "we have to".

3.4 Domain Knowledge

The TRAINS domain requires a relatively modest quantity of knowledge, in terms of basic object and event-types. Still, the interactions of properties and events can get quite complex. Some of the properties of actions and events that must be dealt with even for such a simple domain are:

1. Actions and events take time. During this time, they can have a rich structure, including intermediate states and decomposable sub-actions, yet the activity over that stretch of time is appropriately described as a single event.

2. The relationship between actions and events and their effects is complex. For example, some effects become true at the end of the event and remain true for some time after the event. Other effects only hold while the event is in progress.

3. External changes in the world might occur no matter what actions an agent plans to do, and can interact with the planned actions. Possible external events should be an important factor when reasoning about what effects an action might have. Certain goals can only be accomplished by depending on external events.

4. Actions and events can interact in complex ways when they overlap or occur simultaneously. In some cases, they will interfere with certain effects that would arise if the actions were performed in isolation. In other cases the effects will be additive. And in still other cases, the effect of performing the two actions can be completely different from the effects of each in isolation.

5. Knowledge of the world is necessarily incomplete and unpredictable in detail, thus reasoning about actions and events can only be done on the basis of certain assumptions. No plan is foolproof, and it is important that a formalism make the necessary assumptions explicit so that they can be considered in evaluating plans.

These and similar considerations led us to a representation based on interval temporal logic, the technical details of which are presented in [Allen and Ferguson, 1994]

3.5 Planning Knowledge

In addition to traditional requirements for a problem solving system, such as the ability to represent the problem, the solution, and operators which lead from one to another, the embedding of a plan reasoner within a mixed-initiative dialogue system adds additional requirements and representational challenges.

The first requirement is that the input to the plan reasoner be "language-like." That is, it needs to be expressed in a highly expressive language with multiple levels of representation. These include the domain level (trains, commodities, cities, time, *etc.*), the plan level (plans, actions, events, enablement, generation, *etc.*), and the problem-solving level (strategies, focus of attention, choice-points, *etc.*). All of these and more are candidates for discussion in a system such as TRAINS. In all cases, the objects involved may be only partially or indefinitely described, and part of the job of the plan reasoner may be to determine more fully which objects are involved (*e.g.*, "move a boxcar").

The next requirement is that planning and plan recognition (and other plan reasoning) be interleaved during a dialogue. Plan recognition is typically done to understand the meaning of what has been said, then planning is done to refine the proposal and possibly generate new suggestions, while plan evaluation is, in a sense, a continuous process. Further, the different forms of plan reasoning are performed in different ways. Plan recognition is typically a satisficing process, since we are primarily interested in ensuring coherence of the dialogue. That is, so long as we can make something of the user's statements, we shouldn't expend too many resources trying to optimize the results. Planning, on the other hand, is more of an optimizing process, performed when the system has an opportunity to think about how to improve or complete the plan. In the case of an intelligent planning assistant, of course, the results are expected to be both accurate and helpful.

This interleaved nature leads to the third requirement, which is that it should be possible to perform plan reasoning in arbitrary knowledge contexts. We note below that the dialogue manager maintains belief contexts representing different views of the conversation. The plan reasoner can be called upon to reason in any of these—in a system-related context the system is "thinking for itself," and in a user-related context the system is simulating the thought processes of the user. Because the contents of the contexts depend on what has already been said, the plan reasoner must be able to start with a partial plan in a context and reason from it.

Fourth, the plan reasoner must return useful information – the results of plan reasoning operations need to be expressed declaratively so that they can be used to generate utterances, and update beliefs. Since not everything can be communicated to the user at once, some parts of the information returned by the plan reasoner may not become part of the final plan, at least not immediately. And if a plan reasoning operation fails for some reason, it is essential that useful information describing the failure be returned in order that the system can at least inform the user, if not address the problem at the problem-solving level.

Finally, much of the reasoning performed by the plan reasoner is defeasible. This means making assumptions explicit, being able to assert and retract them, and, importantly, realizing that the system shouldn't always perform complete reasoning. Rather, it can make an assumption or, since the system is interactive, indicate that it would be better to ask the user a question about how to proceed.

3.6 Plan Execution Knowledge

Acting in a world has a slightly different set of requirements from reasoning about and talking about action. In addition to reasoning at an abstract level about change, a plan executor must reason at the level of perceptions and manipulations that it can perform – how it can achieve a desired action and how it can tell whether and when it has been performed. The eyes, ears, and arms of the TRAINS system are the agents in the simulated world. In performing actions in a plan, such as moving a boxcar, the executor must reason in terms of messages that must be sent to an engineer. Moreover, to determine whether the action has succeeded the executor must solicit reports on the current and future state of the world from this engineer and other agents.

In addition, the executor must reason about the information the individual agents have, since these agents only know about their immediate surroundings. Requests for action and information must be formulated in terms that the agents can understand – condition action pairs, with conditions consisting only of an agent's local state. The reports the executor receives back will, likewise, only describe aspects of the reporting agent's local state. The executor must relate the low-level observations back to anticipated events that are important to the plan.

4 The TRAINS-93 KR Formalisms

In starting the project, it was not obvious that any existing KR formalism or inference system would have been appropriate to represent and reason efficiently about all the different kinds of necessary information described in the previous section. Since there were already well-established traditions and tools designed to reason about several of the sub-issues, we decided to use those as starting points adding and adapting where necessary. In addition, adopting different formalisms made it easier to develop general-purpose components that could also be used in isolation. We felt it was best to postpone any unification attempts until we gained a better understanding of the problems we had to face.

4.1 Episodic Logic

The representation used by the Parser to represent the meaning of utterances is called Episodic Logic (EL). EL is an intensional situational logic developed as a semantic and knowledge representation for general NLU [Hwang and Schubert, 1993a; Hwang and Schubert, 1993b]. The most distinctive feature of EL is its natural-language like expressiveness, aimed at meeting our stated requirements of expressive, derivational, and semantic adequacy. It allows for generalized quantifiers, lambda abstraction, predicate and sentence modifiers, predicate and sentence nominalization, intensional predicates (corresponding to wanting, needing, believing, *etc.*), tense and aspect, surface speech act operators, and explicit situational variables together with operators to specify what's true in a situation.

As a result, the correspondence between surface form and logical form (LF) can be kept extremely simple. In general, single-morpheme content words are interpreted as atomic predicates or operators on predicates. Each of the phrase structure rules in the TRAINS grammar is paired with a simple compositional semantic rule, and the Parser computes the initial, underspecified EL representation of an utterance as it builds a parse tree for the utterance. The initial representation is underspecified in that it may contain unscoped operators (quantifiers, tense operators), indexical expressions (*e.g.*, tense operators, pronouns) and ambiguous predicates. An example of a grammar rule is

rule name: P2_Ploc+N2
syntactic rule: ((*P 2bar*) (*P 0bar loc*) (*N 2bar acc object*))
semantic rule: (*:p 1 2*) .

In the syntactic rule, the first constituent (P 2bar) is the left-hand side and the remaining two constituents the right-hand side of a GPSG-like rule. Thus the rule admits the formation of a locative prepositional phrase from a locative preposition and an accusative noun phrase, like *at Corning* from *at* and *Corning.* (The rule name concisely echoes the rule.) The *loc* feature is transmitted from the preposition to the prepositional phrase through the head feature principle of GPSG [Gazdar et al., 1985]. Unlike GPSG, our grammar uses hierarchies of atomic features. For instance, *loc* belongs to the set of features {*loc, time,* *phow, pwhy, subj-matr, oppos*} which subdivide the *ppred* (prepositional predicate) feature, and *loc* is in turn subdivided into {*place, path*}. The lower-level features unify with their hierarchy ancestors. Such feature systems, in conjunction with feature percolation principles, allow for quite compact representations of auxiliary verb structure, noun premodification, adverbial modification, subcategorization, *etc.* The semantic rule in the example indicates that the interpretation of the prepositional phrase is obtained by applying the interpretation of the first constituent (a dyadic predicate) to the interpretation of the second constituent (*e.g.*, the individual denoted by the NP, if the NP is referential). The keyword *:p* denotes predicate application. Additional examples will be seen in section 5, as annotations of the parser output.

While all formulas of EL may be written in lisp-like prefix form, complete sentential formulas are usually written in infix form, with the subject preceding the predicate as in English. This infix form is distinguished by the use of square brackets, *e.g.*, for *Train TR1 is at Avon* we might write

(*pres* [*TR1 loc-at Avon*]).

(In the "lispified" version of the infix notation we would use (i: ...) for [...], as will be seen in section 5.) The *pres* operator heading this formula is indexical, and is replaced in the deindexing process by predications involving explicit episodic variables.

EL makes extensive use of predicate and sentence modifiers: operators that map predicates into other predicates or sentence meanings into other sentence meanings. For example, the predicate *OJ* is true of quantities of orange juice. The predicate operator *make* takes this predicate and produces another predicate (*make OJ*), which is true of any agent who makes orange juice. Many different syntactic constructs have simple interpretations when viewed as predicate modifiers, including attributive adjectives, certain classes of adverbs, and certain kinds of intensional verbs (like *make*, above). Simple examples of sentence modifiers are modal ones such as *maybe* and *should.* Among the most important predicate and sentence modifiers are those corresponding to PP adverbials. For instance, the phrase *go to Dansville* is represented as

((*ADV-A* (*to-loc Dansville*)) *go*).

Here *ADV-A* is an operator that transforms a monadic predicate (viz., (*to-loc Dansville*)) into a predicate modifier (operating on *go*). Similarly, the sentence *Train TR1 left at 8 PM* would be represented by the form

((*ADV-E* (*at-time 8PM*)) [*TR1 leave*]),

where the *ADV-E* operator converts a monadic predicate (in this case, (*at-time 8pm*)) into a sentence modifier. Like the tense operators *pres* and *past*, adverbial predicate and sentence operators are replaced in the deindexing process by predications explicitly involving episodic variables.

EL also has operators for predicate and sentence nominalization (reification). The main predicate nominalization operator, *K*, constructs a kind or type of object from a predicate. The noun phrase *orange juice*, for example, often refers to orange juice in general rather than to some specific quantity of orange juice, as in the sentence *I like orange juice.* If *OJ* is a predicate true of quantities of orange juice, then (*K OJ*) is the kind of thing, or substance, orange juice. Kind terms are used extensively with events as well, as in the sentence *I want to load the oranges*, where what is desired is the performance of some action of the kind "load the oranges," rather than a specific, existing act of loading oranges. Sentence nominalization operators are crucial to the EL representation of attitudes. For instance, to express that *The user told the system that engine TR1 is at Avon*, we would form the nominalized sentence (or proposition)

$$(\mathit{That}\ (\mathit{pres}\ [\mathit{TR1}\ \mathit{at\text{-}loc}\ \mathit{Avon}])),$$

and this would become the object of a *tell*-predication, whose subject is the user.

The forms of quantifiers and of lambda abstracts in EL can both be illustrated with the interpretation of the phrase *the engine at Avon* (utterance 9-13 in the sample dialogue):

$$\langle \mathit{The}\ \lambda x[[x\ \mathit{engine}] \wedge [x\ \mathit{at\text{-}loc}\ \mathit{Avon}]]\rangle$$

This expression could in principle be directly interpreted as a generalized quantifier, but in EL it is instead viewed as an unscoped quantifier, yet to be "raised" so that it will have a sentence as its scope. The alternative ways of raising unscoped quantifiers (and certain other unscoped operators) account for scope ambiguities. In the present case, if we consider a little more of the utterance embedding the quantifier, *we should move the engine at Avon to Dansville*, a possible scoping of the corresponding EL formula is

$$\begin{array}{l}(\mathit{The}\ x\text{:}\ [[x\ \mathit{engine}] \wedge [x\ \mathit{at\text{-}loc}\ \mathit{Avon}]] \\ \qquad\qquad (\mathit{pres}\ (\mathit{should}\ [\mathit{We}\ ((\mathit{ADV\text{-}A}\ (\mathit{to\text{-}loc}\ \mathit{Dansville}))\ (\mathit{move}\ x))]))).\end{array}$$

Note that the general form of scoped quantification is $(Q\alpha : \phi\,\psi)$, where Q is the quantifier, α is a variable, ϕ is an open sentence restricting the domain of the variable, and ψ is the sentence comprising the scope of the quantifier.

The deindexed EL representation of sentences describing events, processes or situations contains explicit *episodic* variables. Like situations in situation semantics [Barwise and Perry, 1983], episodes are partial states of affairs in the world, encompassing some properties of some participants over some period of time at some location. Episodes subsume Davidsonian events [Davidson, 1967] as used in many representations, *i.e.*, an event is a particular kind of episode. The content of episodes can be described using two special operators, * and **.

The ** operator indicates that an arbitrary formula *characterizes* an episode. A simple example would be the formula

$$[\mathit{TR1}\ \mathit{move}\ \mathit{Avon}\ \mathit{Dansville}]\ {**}\ \mathit{E1}$$

which is true if episode $E1$ is characterized by engine *TR1* moving from Avon to Dansville. In a Davidson-based event logic, this might be written as *move(E1, TR1, Avon, Dansville)*, where the event is an argument of the predicate. But EL can represent much more complex events than is possible using event arguments of predicates. For example, the "negative situation" $E2$ of train *TR1* failing to arrive on time can be characterized as follows (ignoring the details of "arriving on time"):

$$[\neg[\mathit{TR1}\ \mathit{arrive\text{-}on\text{-}time}]]\ {**}\ \mathit{E2}.$$

A Davidsonian representation could approximate the characterization of $E2$ by saying that there is no event of *TR1* arriving on time *during* E2, but such an approximation is inadequate for capturing *causal* relations. For example, it may be true that some event, $E3$, *during* which there was no instance of *TR1* arriving on time caused *TR1* to derail – for instance, $E3$

could be the event of *TR1* taking a curved section of track too quickly. On a Davidsonian analysis of negative situations, this causal relation can be seen to entail, incorrectly, that the failure of *TR1* to arrive on time caused its derailment![3]

As a quantified example, the episode $E2$ consisting of every train going to some city could be characterized as follows:

$$(\forall t : [t\ \mathit{train}]\ (\exists x : [x\ \mathit{city}]\ [t\ ((\mathit{ADV\text{-}A}\ (\mathit{to\text{-}loc}\ x))\ \mathit{move}])\ {**}\ \mathit{E2}.$$

This could not easily be encoded as a single event in an event-based logic.

The second episodic operator provided by EL, $*$, asserts that a proposition is true in an episode. This is a weakening of the $**$ operator, not requiring the proposition to characterize the episode as a whole, only some aspect of it. In that sense $*$ is analogous to the $\models$ relation between a situation and an infon in situation semantics. Both the $*$ and the $**$ operator are important in EL axioms and inference about causal relations and other relations between episodes [Hwang and Schubert, 1993b].

A semantically adequate knowledge representation with all of the above syntactic machinery naturally requires a rich ontology, *e.g.*, to accommodate intensional predicate and sentence modifiers, the abstract kinds, properties and propositional entities formed by nominalizations operators, and the episodes, times and locations referred to in episode descriptions. EL allows for all of these, as well as for numbers, collections and sequences. All entities in the domain are regarded as "possible individuals", and this includes possible episodes (situations), possible facts (which correspond to consistent propositions), *etc.* Intensionality is modeled by interpreting atomic symbols in a situation-dependent way. For example, the two-place predicate *leave* is interpreted as a partial function

$$f\colon \mathrm{D} \rightarrow (\mathrm{D} \rightarrow (\mathrm{S} \rightarrow \{0,1\})),$$

where D is the domain of possible individuals and S is the set of possible episodes (situations), a subset of D. It can be seen that by applying f to two successive individual arguments, we obtain a sentence intension, a partial function from possible situations to the truth values $\{0,1\}$. The fact that such functions are partial allows for "fine-grained" semantic distinctions between propositions that are logically equivalent in the classical sense. Also, by making the situational argument in predicate intensions (such as f here) the last argument (rather than the first, as in Kripke- or Montague-style semantics), we are able to simplify our lexical and phrasal semantic rules, avoiding Montague's ubiquitous intension/ extension operators. The interpretation of the * operator within this framework corresponds closely to truth in a situation; *i.e.*, $[\phi * \eta]$ is a way of saying that the semantic value of ϕ at the situation denoted by η is 1 (truth). The conditions for truth of $[\phi\ {**}\ \eta]$ are more stringent, requiring that η "as a whole" be a ϕ-episode, rather than just having a ϕ-episode as some part or aspect of it. The formal distinction relies on "persistence" of sentential

[3] More exactly, $\mathit{speed\text{-}around\text{-}curve}(E3,TR1) \wedge \mathit{derail}(E4,TR1) \wedge \mathit{cause\text{-}of}(E3,E4) \wedge \neg(\exists e)\mathit{during}(e,E3) \wedge \mathit{arrive\text{-}on\text{-}time}(e,TR1)$ entails $(\exists e')[\neg(\exists e)\mathit{during}(e,e') \wedge \mathit{arrive\text{-}on\text{-}time}(e,TR1)] \wedge \mathit{derail}(E4,TR1) \wedge \mathit{cause\text{-}of}(e',E4)$. This entailment, obtained by existential generalization of $E3$, says that some eventuality e' during which $TR1$ did not arrive on time caused the derailment $E4$. Thus if we accept "some eventuality e' during which $TR1$ did not arrive on time" as a characterization of a negative eventuality (an eventuality of $TR1$ failing to arrive on time), we are led to the unwarranted conclusion that an eventuality of $TR1$ failing to arrive on time caused its derailment

truth in a set of semilattices of situations, each having a maximal possible situation – a possible world – as its "top".

EL was originally designed for story understanding, but its comprehensiveness and its NL-like syntax and semantics made it an entirely plausible candidate for dialogue interpretation. In fact, we encountered no fundamental difficulties in this redeployment. The real challenges lay in expanding the very small syntactico-semantic fragments we had built to deal with a few narrative sentences, so as to cover the selected TRAINS dialogues in a principled way. We needed to develop a system of several hundred features (organized into some 27 feature hierarchies) and ways of assigning default lexical features; fairly broad-coverage syntactic rules (including many types of questions and imperatives, coordination. *there*-insertion, subcategorization patterns, *etc.*); and most of all, carefully thought-out solutions to numerous specific semantic problems. For example, the question arose whether to treat modals such as *should, shall* and *better* (see utterance 1 of the dialogue) as sentence or predicate modifiers at the level of semantics. We chose the former so as to be able to account for "opaque" readings of sentences such as *Someone should pick up the oranges*, or *Somebody better be there when the shipment arrives.*

Some of the simplest utterances – yes/no, right, okay – proved surprisingly subtle and controversial. The problem is that these utterances have implicit arguments – *some* salient propositions or proposal is being affirmed / denied / approved. The question is whether to make these implicit arguments explicit in the immediate LF, or to send unanalyzed atoms to the deindexing and discourse modules. In the case of *right* and *okay* we opted for explicit arguments because they can occur with subjects (*e.g.*, *That's right*) and be viewed as members of a productive group of APs and other predicate phrases – consider, *true; wrong; fine; very good; no problem; well done;* etc. *Yes/no* could be handled either way. Of course in all cases the work of identifying the implicit referents is left to deindexing and discourse processing. Similarly *so* is a subtle cue word establishing a relevance or following-from connection to *something* prior, and in this case we simply translate this as an initial coordinator (a sentence modifier) whose significance is a matter for the discourse module to consider.

Perhaps one elaboration of the original narrative understanding framework particularly worth mentioning is the addition of *locutionary acts* (surface speech acts) to the interpretations of utterances. *I.e.,* the LF of a surface declarative contains the top-level construct (ignoring the explicitly added utterance episode)

[*Speaker tell Hearer* (*That* ...)],

for questions we have an *ask* act, for imperatives an *instruct* act or something similar, *etc.* This locutionary act information is very important for subsequent reasoning about discourse structure (in contrast with the case of narratives). Technically, the locutionary acts are introduced into the LF's through the terminal symbols of utterances, *i.e.*, punctuation, turn-taking or (ultimately) prosodic markers.

There were many additional problematic issues concerning logical form, including the semantics of *problem, plan, make*, purpose clauses, proper nouns, numbers and measures, *etc.* None of these problems were trivial, but the important point is that the EL framework provided sufficient flexibility to allow us to consider various plausible options, and to implement workable solutions. (For some further remarks on *plans*, see Section 6.3.)

The main shortcoming of EL as a representation for conversations is that it focuses primarily on utterance *content*, not *context*. In [Hwang and Schubert, 1993b] context structures called *tense trees* and an algorithm for tense deindexing are proposed. However, this algorithm presupposes that tense operators and quantifiers have already been correctly scoped, and that referent determination can be done independently of scoping and tense deindexing. For the TRAINS system, we needed an integrated account of these processes.

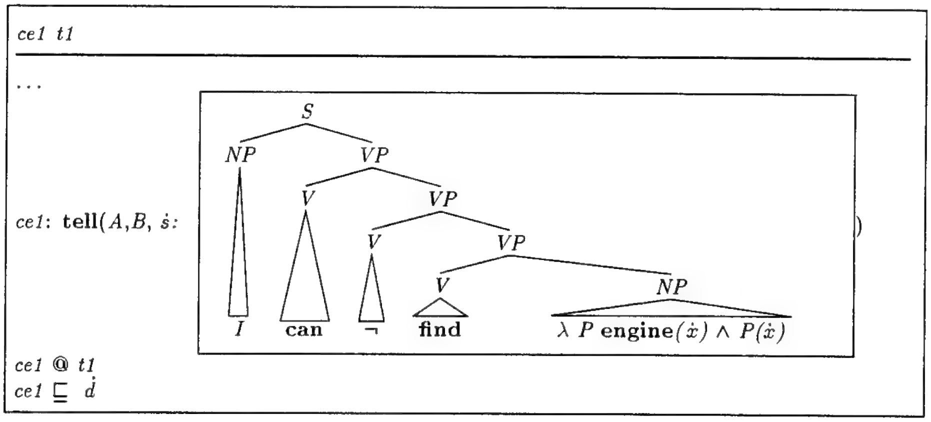
4.2 CRT

Conversation Representation Theory (CRT) [Poesio, 1994] is an account of contextual interpretation and ambiguity resolution in conversations that builds on the theory of lexical semantics developed in Episodic Logic. The formalism used in CRT extends the language of EL with *underspecified expressions*—expressions that indicate which aspects of a sentence's meaning have to be disambiguated and are given a proper semantics [Poesio, 1991; Alshawi and Crouch, 1992; Reyle, 1993; Poesio, 1995]. The theory also includes an account of disambiguation as defeasible reasoning over underspecified interpretations of the discourse situation.

One of the two main assumptions motivating CRT is that the interpretation of a sentence is obtained by means of inferences that crucially involve underspecified representations. The underspecified representations produced by the parser (see previous section), which in EL were given only an implicit interpretation in terms of their disambiguation, are seen as first-class objects of the language of CRT, and disambiguation is formulated as a process of inference over such representations. A second important assumption, borrowed from work on discourse in the AI tradition such as [Grosz and Sidner, 1986], is that the process of disambiguation crucially involves information which is pragmatic in nature, such as the hierarchical relations between speech acts. Thus, the model of context adopted in CRT, while inspired in many ways from those used in 'dynamic' theories of context such as Discourse Representation Theory (DRT) [Kamp, 1981; Kamp and Reyle, 1993] and Dynamic Predicate Logic [Groenendijk and Stokhof, 1991], is a model of the effect that utterances have on the discourse situation, rather than of the effect that sentences have on the propositional content of a text.

For an example of disambiguation, consider the sentence *I can't find the engine.* Disambiguation is initiated when the context is updated with an underspecified interpretation of an utterance. Interpreting this sentence in a particular context involves recognizing the speech act performed by the speaker, the intended referent of *the engine*, and the scope of the modal operator **can** and the negation operator. The effect of updating the existing context after an utterance of this sentence is represented by the following CRT expression.

(1)



The expression in (1) can be paraphrased as: a speech act *ce1* occurred, of A telling B about situation $\dot{s}$ that $\dot{s}$ is characterized by the underspecified expression within the nested box. (In what follows, we will also use the term *conversational event* to refer to speech acts.) The following things have to be explained about the expression in (1). First of all, context is modeled as a *discourse representation structure* (DRS), a pair consisting of a set of *discourse markers* (here, *ce1* and *t1*) and a set of *conditions* (wffs). DRSs can be nested; the 'top' DRS, which provides a representation of the common ground as a whole, is called 'Root DRS'. In Discourse Representation Theory, the root DRS provides a representation of the 'described situation', or the situation the discourse is about; in CRT, the root DRS is a representation of the *discourse situation*, i.e., the situation in which the participants to the conversation find themselves. The description of the described situation (here, $\dot{s}$) is embedded in the description of the discourse situation.

Secondly, the representation in (1) is underspecified in many ways. As in the EL interpretation of complete utterances, what is initially recorded in the model of context is the occurrence of a locutionary act (here, **tell**), from which the actual intentions of the speaker have to be recovered (see following sections). The expressions of the CRT language used to encode alternative scope readings are syntactic trees whose leaves are EL expressions. This representation derives from the unscoped LF of EL, but makes it more explicit that an 'unscoped representation' is simply one in which operators have been left in their syntactic position. These expressions are another aspect of an utterance's interpretation that has to be resolved in context.

The third aspect of the interpretation to be resolved in context is the value of anaphoric expressions, called *parameters* and written down as variables with a 'dot': $\dot{x}$. These are aspects of an utterance's meaning which have to be related ('anchored') to some previously introduced element of context (discourse marker). In (1) there are three such parameters: the described situation $\dot{s}$, the referent of the definite description $\dot{x}$, and the *conversational thread* $\dot{d}$. This last bit needs some explanation. Each speech act is assumed to be part of a 'conversational thread', which is a particular type of situation. All conversational events in the same thread are assumed to be 'about' the same *discourse topic*, which is another situation. Conversational threads are used in CRT to capture Grosz and Sidner's notion of discourse segment, whereas discourse topics are a formalization of the notion of 'focus space'. Thus, whereas in Grosz and Sidner's theory we would have a focus space extending another focus space on a stack, in our framework we have a situation extending another situation; in this way we do not introduce an additional stack mechanism, and the processes of focus space construction and focus space shift can be modeled as reasoning processes, as discussed in [Poesio, 1993].

The rest of the syntax of the language used in CRT is borrowed from EL: thus we have lambda expressions, modifiers, kind-forming operators, and all the other tools discussed in the previous section. One remaining syntactic difference from EL is the use of expressions of the form $s{:}\Phi$ instead of $[\Phi * s]$ to assert that Φ holds of s. The difference is not just a matter of syntactic sugar, as discussed below.

Because of the need to give a direct interpretation to underspecified expressions, the semantics of CRT is a generalization of the semantics of EL. EL expressions denote functions from situations to objects in the domain (for example, the denotation of a sentence is a function from situations to truth values); the expressions of the CRT language, instead, denote *sets* of such functions; for example, the denotation of a sentence is a set of functions from situations to truth values. The resulting logic is much weaker than the one of EL.[4] (This is one reason why disambiguation is done by defeasible reasoning.) DRSs are given the same type of sentential expressions, i.e., they are interpreted as (sets of) functions from situations to truth values (situation types).

The semantics of the 'semantically annotated syntactic trees' used to capture scopal underspecification is defined bottom-up using a 'CV' function (for Cooper Value) that assigns to each of them as a value a set of sequences of expressions $\{\langle\alpha_1, ...\rangle, \langle\beta_1, ...\rangle, ...\}$. The first element of each sequence provides a phrasal LF of the appropriate type (*e.g.*, a variable or generalized quantifier for an NP, a predicate for a VP, and a wff for an S), while the remaining elements are quantifiers or other operators "in storage". For example, the CV of 'find an engine' would be a set of two sequences, $\{\langle\lambda x\ \exists y\ \mathit{engine}(y) \wedge \mathit{find}(x, y)\rangle, \langle\lambda x\ \mathit{find}(x, i), \langle i, \lambda P\ \mathit{engine}(y)\ \wedge\ P(y)\rangle\rangle\}$. The denotation of a tree [S ...] is then the set of functions that can be "extracted" from CV([S ...]) (each of whose elements is required to be a singleton sequence, *i.e.*, there are no elements "in storage").

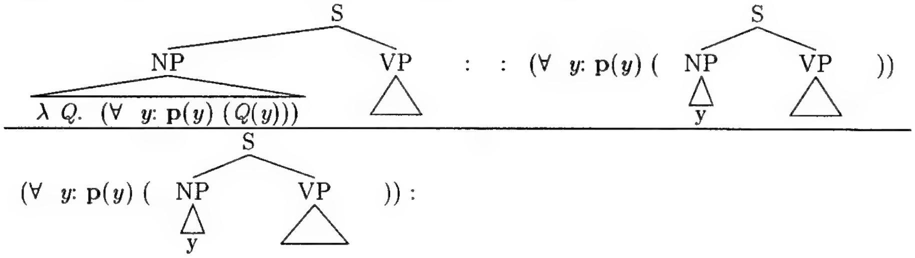
A detail of importance when talking about the anaphoric accessibility of discourse markers, is that expressions in CRT are assigned a value with respect to a situation, a variable assignment, and a function C ("cases") from situations to variable assignments. Conditions like $s{:}\mathrm{K}$ assert that a situation s is of the type specified by the DRS K, and in addition they shift the parameters of evaluation so that the value of the discourse markers occurring in K is provided by the variable assignment associated with the value of s, *i.e.*, $\mathrm{C}([\![s]\!])$, (where $[\![s]\!]$ is the semantic value of s with respect to the original variable assignment and the original cases C). In this way, the discourse markers introduced in a statement about situation s are accessible when interpreting an anaphoric expression which is part of a second statement about the same situation.

The disambiguation rules are formulated as inference rules which result in a less underspecified interpretation. The reasoning framework in which inference is formulated resembles Prioritized Default Logic [Brewka, 1991] in that the rules of disambiguation are Reiter-like default inference rules with different priorities, and in that the result of disam-

[4] For instance, $\Phi \vee \neg\Phi$ is not a theorem.

biguation are zero, one or more extensions of the default theory consisting of the context augmented with the initial interpretation of the utterance together with the set of default inference rules. Space prevents a discussion of the rules actually used in the system; as a simple example of what a disambiguation rule might look like, here is how one could formulate a hypothetical rule assigning wide scope to operators in subject position. The rule is a "normal default rule" [Reiter, 1980]. Notice how the result of disambiguation is a partially disambiguated CRT expression, with the universal quantifier taking scope over whatever operator occurs in the VP.

GRAMMATICAL-FUNCTION-PRINCIPLE



The disambiguation process is subject to a *Condition on Discourse Interpretation* which, roughly speaking, states that an extension cannot contain underspecified expressions. The result of interpretation is therefore an expression which can also be given a semantics within Episodic Logic; this makes it possible to translate such expressions into the other formalisms used in TRAINS.[5] For example, the following expression is the final result of the disambiguation of the sentence *I can't find the engine*:

(2)

ce1 t1 s

. . .

s: **engine**(x) **Avon**(w) **at**(x,w)

ce1: **tell**(A,B, *s:* ¬ **can**(**find**(A,x)))

ce1 @ *t1*

ce1 ⊑ *ct1*

Notice that the described situation of the conversational event has been identified as the previously introduced situation *s*, that the scope of the operators has been resolved, and that the conversational thread to which *ce1* belongs has been identified. The results of speech act interpretation are not presented in (2); these would be expressed as the introduction of a new speech act *ce2* generated by *ce1* (in the sense of Goldman [Goldman, 1970], *i.e.*, occurrence of *ce1* by its nature involves the occurrence of speech act *ce2*).

[5] At the moment, the possibility of such a translation is merely a conjecture based on our experience in building the interface to EBTL - no formal result has been proved.

The 'Scope And Deindexing' module of TRAINS-93, SAD-93, is a partial implementation of Conversation Representation Theory. The system consists of an implementation of a prioritized default reasoner, together with declarative rules that encode theories of disambiguation for scope, definite description interpretation, pronoun interpretation, modals, and tense, among others. The input to the system is the underspecified representation produced by the parser; the output is a set of hypotheses about possible interpretations. The 1993 implementation was tested on both the dialogue of Figure 2 and an entirely different, shorter dialogue used in the TRAINS-91 system.

4.3 EBTL

EBTL (Event Based Temporal Logic) is used by the Speech Act Interpreter, Dialogue Manager and Plan Reasoner, and as a communication language between these modules. It is a typed logic based on interval temporal logic [Allen, 1984], which was designed to facilitate reasoning about actions and events in temporally complex worlds. EBTL is a more restricted language than EL or CRT: only events and some simple states can be represented directly rather than general episodes and situations.

In starting the TRAINS project, the RHET knowledge representation system [Allen and Miller, 1991] seemed a good starting point for representing the plan and domain knowledge, having been used and developed in parallel with previous discourse [Allen et al., 1989], temporal reasoning [Koomen, 1990], and plan recognition efforts [Miller, 1990] at Rochester. RHET provides theorem-proving (both forward and backward chaining inference), as well as type hierarchies and typed unification, equality and inequality reasoning, temporal reasoning, frame-like event representations, and an explicit hierarchical context mechanism for representing belief contexts, hypothetical reasoning and other modalities.

It quickly became apparent, however, that additional features were needed for the TRAINS system reasoning functions. EBTL provides additional constructs that proved essential for representing the meanings of natural language input and relations of facts to plans, as well as a more straight-forward Lisp interface. These include explicit quantifiers, discourse markers, lambda expressions and quoted propositions. The underlying inference is performed by RHET, augmented with special reasoners in Lisp for the new constructs.

The terms in EBTL are those allowed in RHET. Constants are written as atoms within square brackets. Thus `[ENG3]` is a term. Distinct names do not necessarily identify distinct objects, since the language supports equality reasoning. When terms are defined, their type is usually defined as specifically as possible. `[ENG3]`, for instance, is declared to be of type `T-ENGINE`. Variables are indicated using the form ?*varname***typename*. For instance, `?x*T-ACTION` is a variable that ranges over all objects of type `T-ACTION`.

Propositions in EBTL are represented by lists. For instance, since `EQ?` is the equality predicate, the proposition `(EQ? [F-ENGINE [M2]] [ENG3])` would assert that `ENG3` is the engine causing the move event `M2`.

EBTL has a reified event representation, where event instances are denoted by explicit terms in the language. Figure 4 shows part of the event hierarchy for the TRAINS domain. This hierarchy predominantly deals with events caused by actions, which capture different actions available to the TRAINS domain planner. Events that cannot be directly executed are classified as non-action events. These events are necessary to provide a clean interface with the natural language system, because people often talk in terms of the events caused rather than the actions executed. For example, the system never plans to execute an arrive event. Rather, it plans a move action, one aspect of which is sometimes described as an arrive event. The specific features of events are identified by role functions, which are defined over event classes. For example, all actions will have agent roles, as well as the general time role for all events. Subactions such as `T-MOVE` will also have (in the TRAINS-93 domain) an engine role, *etc.*

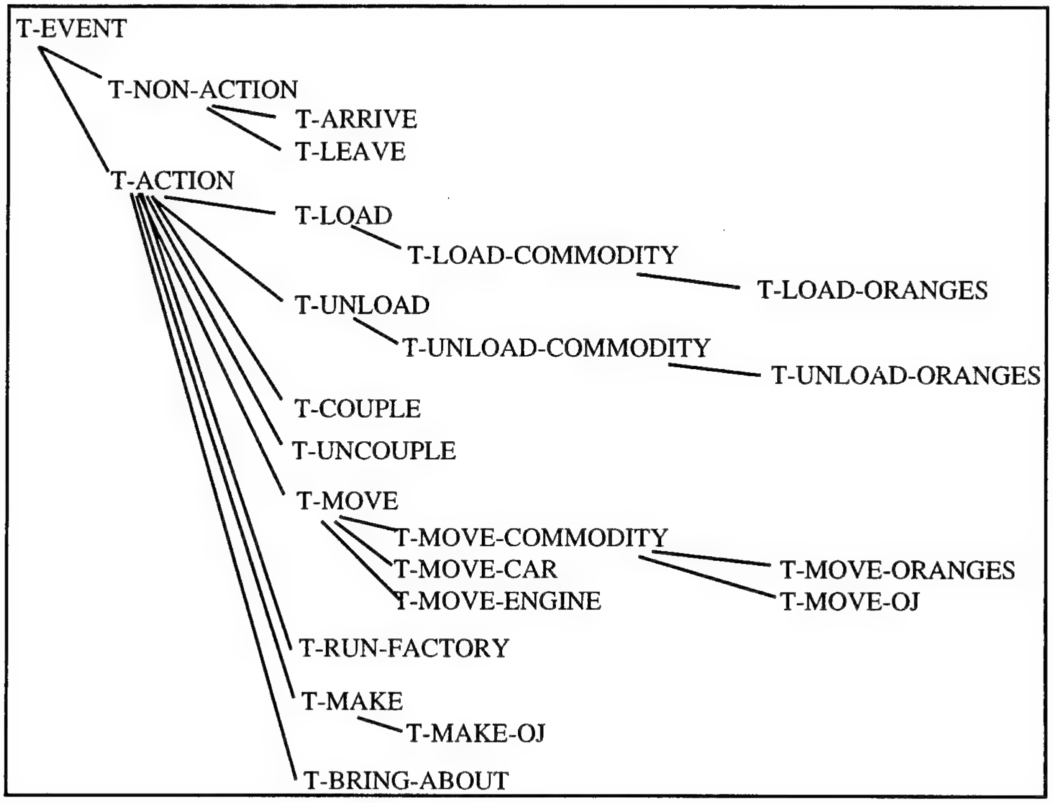


Figure 4: The event hierarchy of TRAINS domain events

Conversation Level Representation

EBTL also includes features for reasoning about conversational interpretation. These include: a subclass of the action type from Figure 4 for speech acts, operators for representing combinations of primitive speech acts (*e.g.*, conjunction, exclusive and non-exclusive disjunction, and sequencing), relationships between conversational events (*e.g.*, preceding or following in a discourse segment), other conversation act types (as described in [Traum and Hinkelman, 1992]), including predicates for argumentation relations between speech acts, turn-taking, and grounding acts.

The core speech act types used in the TRAINS-93 system are shown below, with their meanings:

T-INFORM The speaker provides new information in order to establish a shared belief in the proposition asserted.

T-YNQ The speaker asks a yes-no question, creating an obligation for the hearer to respond.

T-CHECK The speaker is attempting to verify that a certain proposition is true.

T-SUGGEST The speaker proposes a new item (action, proposition) as part of the plan.

T-REQUEST The speaker aims to get the hearer to perform some action. In the TRAINS domain, this is treated like a suggest, with the addition of an obligation on the hearer to respond.

T-ACCEPT The speaker agrees to a prior proposal by the hearer.

T-REJECT The speaker rejects a prior proposal by the hearer.

T-SUPP-INF The speaker provides additional information (perhaps already known) that augments, or helps the hearer interpret some other accompanying speech act.

T-SUPP-SUG The speaker makes a supplementary suggestion of content, which is presupposed to be part of the plan by other accompanying suggestion or request.

Many of the types of discourse structure and mental state needed by the dialogue manager (such as the discourse segments, turn-taking, and grounding structures mentioned in Section 3.2) are represented not within EBTL itself but as RHET belief contexts or lisp data structures containing EBTL formulae. However there are EBTL predicates which can be proved by functional calls to inspect these data structures. More on the conversation-level representation can be found in [Traum, 1994].

EBTL also contains terms that explicitly denote plans and predicates that are used to describe the content of plans. Some examples of these are:

`(GOAL-OF` *proposition plan*`)`: the proposition is a goal of the indicated plan

`(USES` *object proposition plan*`)`: the object is used in the plan.

`(EVENT-IN` *event plan*`)`: the event is part of the plan.

These predicates are used by the dialogue manager to treat plans in an abstract way – objects with particular properties. Different combinations of these properties may be present in different mental modalities, which will provide the specific motivation for the system to perform plan related utterances such as suggestions and acceptances. These predicates are also used to represent the contents of plan related speech acts, and as calls to the domain plan reasoner to incorporate these items into a partial plan.

Plans Graphs

In addition to describing the properties of plans using EBTL plan predicates, the TRAINS-93 domain plan reasoner maintains an additional representation of plans as connected, directed, acyclic graphs of *event* and *fact* nodes labeled by events and propositions, respectively. These plan graphs are given a formal semantics in terms of *arguments*: they represent an argument that a certain course of action under certain conditions will achieve certain goals. Since plans are objects in the ontology, plan description predicates can be defined that impose constraints on the plan-as-argument. There are *structural* predicates, such as `ActionIn`, `Enables`, `Premise`, `Goal`, *etc.*, that are defined in terms of the structure of the plan graph. Then there are *evaluation* predicates. These include *absolute* predicates such as `Plausible` or `Impossible` and *relative* predicates that allow plans to be compared, for example, according to resource usage or time constraints. Further details are available in [Ferguson, 1995].

4.4 Tyro

In order to do the reasoning necessary for executing and monitoring plans and statistical reasoning about the effects of actions, we have developed a language, called Tyro. Like EBTL, Tyro is implemented as an extension of RHET. Tyro allows representation of statistics about the number of observations of event instances of a particular event type and provides techniques for computing confidence intervals from the statistics. It also allows one to specify utility and to apply decision theory based on these utilities and confidence intervals.

Tyro represents individual event instances (e.g., `[Move-Boxcar-1]`) and it reifies sets of event instances specified by lambda abstractions into event types (e.g.,`[Boxcar-Moved]`). Here, the term `[Boxcar-Moved]` represents the set of event instances in which any boxcar was moved; `[Move-Boxcar-1]` represents a particular instance in which a particular boxcar was moved.

Tyro also includes statistical predicates, which are defined over event types. These predicates are `Occurrences`, and `Probability`. `Occurrence` takes a temporal interval, an integer and a set of event types, as in (3); it indicates that the integer is the number of event instances in the intersection of the event types that occured during the temporal interval. `Probability` takes a confidence level, two event types, a probability measure, and a temporal interval as in (4); it indicates that the probability measure is the conditional probability during the temporal interval of the occurrence of the first event given the occurrence of the second event.

(3) `[Occurrence t1 5 [Requested-Boxcar-Move] [Boxcar-Moved]]`

(4) `[Probability .95 [Requested-Boxcar-Move] [Boxcar-Moved] [0.24 0.76] t1]`

In Tyro, a probability measure can be either a point (a number between 0 and 1) or an interval (where both end points are numbers between 0 and 1). In addition, probabilities can either be asserted or inferred from the `Occurrence` predicate. Probabilities that are asserted are entered before the system runs; probabilities that are inferred are calculated after the system has made some observations. A point probability indicates that a programmer asserted a number to be the probability and that the system has insufficient evidence to doubt the assertion. An interval probability indicates either that the programmer asserted an interval probability or that the system has sufficient evidence to conclude that an asserted probability is incorrect.

Tyro infers the probability of a particular event, say [Boxcar-Moved] given a trial event type, say [Requested-Boxcar-Move] by considering the event type that represents the intersection of the instances in each event type, i.e. [Requested-Boxcar-Move-and-Boxcar-Moved] as the success event type and the trial event type [Requested-Boxcar-Move]. It then calculates the confidence for the binomial random variable [Bickel and Doksum, 1977] where an event instance in the intersection event type is a success and an event instance in the trial event type is a trial. If an asserted probability does not fall entirely within this confidence interval, it rejects an asserted probability and concludes that the probability is the confidence interval computed from the number of occurrence of the event types it has seen. If no probability was asserted and no observations have been made, Tyro uses the probability interval [0, 1]. This treatment of probability is based on Kyburg's evidential probability [Kyburg, 1991].

Tyro also represents the utility of event types so it can make decisions based on expected utility. The predicate, `Utility` takes an event type, an integer and a temporal interval, as shown in (5); it indicates that the integer is the utility of the event type during the temporal interval. Utility expresses the desirability of an event; probability expresses the likelihood of an event. Both types of information are necessary to make a choice between events. This difference is reflected in the meaning of the temporal interval associated with utility and probability. The temporal interval associated with probability represents the time over which the evidence for that probability was gathered; the temporal interval associated with utility represents the time over which the event is desirable. For example, If one is shipping oranges, it may be desirable to wait for a train that is scheduled to arrive in the future, but undesirable to wait for a train that has already left. The desirability can be changed by evidence of the effectiveness of the act of shipping the oranges by this train. If, in the past, it has proved impossible to load the oranges, perhaps because the train does not stop at this station long enough, it may not be desirable to wait for the train even if it is scheduled to arrive in the future.

(5) `[Utility [Boxcar-Moved] 10 t2]`

Tyro calculates expected utility by multiplying the probability by the utility – resulting, when the probability is an interval, in an interval expected utility. When an interval expected utility is calculated, there may be no clear choice between alternatives. When there is no choice, one can do one of three things: one may reduce the confidence level in the calculation resulting in a narrower interval, one may avoid making the choice hoping that further information will make the choice clearer, or one may gather more information about needed probability. In Tyro, when one reduces confidence to zero, it calculates a point probability from the maximum likelihood estimate [Bickel and Doksum, 1977]. If one avoids the choice, subsequent events may rule out some actions, making the choice clearer. Because more evidence also narrows the interval, collecting more evidence may also make possible a choice.

Although both EBTL and Tyro allow reasoning about events and event types, these are used for different purposes in the two languages. When planning, events are terms that represent occurrences that may be added or subtracted from the plan as necessary. But when the plan is being executed the event represents something that cannot be modified, but only described. Some of the unspecified properties of an planned event might be important when the plan is executed. For example, whether the axle on `[Boxcar-1]` is broken might be ignored when planning if broken axles are rare enough, but the state of the axle will be crucial to the event that actually occurs. This state of affairs will be discovered only after the event that is purportedly the `[Move-Boxcar-1]` event has occurred but, because the effects of `[Move-Boxcar-1]` do not hold, the plans goals will not hold.

To deal with these problems, the executor views the EBTL event `[Move-Boxcar-1]` as an event type that includes all of the event instances that have the properties specified for `[Move-Boxcar-1]`. Using Tyro, the executor reasons about what it can do to make an event instance occur that will be a random member of the event type corresponding to `[Move-Boxcar-1]`. By reifying the event types, Tyro allows the executor to use statistical probability to choose a random member of the action event type that is most likely to co-occur with the event type corresponding to `[Move-Boxcar-1]`. In addition, Tyro allows the executor to reason that an event type corresponding to `[Move-Boxcar-1]` occurred, but that the instance of this event type that occurred does not support the plan. That is, even though what occurred satisfied all of the conditions specified of `[Move-Boxcar-1]`, it is invalid to assume that the boxcar moved because, in fact, the boxcar did not move. The executor may or may not know why the boxcar did not move, but, being in a position to keep track of the plan, it can determine that the boxcar did indeed fail to move.

5 A Dialogue Processing Example

While a true sense of the operation of the representation systems presented in the previous section can only be achieved by examining a substantial section of dialogue, there is not room here for a complete look at even one of the formalisms on a very substantial amount of dialogue. Instead, we illustrate the use of the implemented KR systems by means of a single example exchange from the conversation in Figure 2, sentences `7.1-7.2` through `8`, repeated here as (6), for convenience. More details on how the rest of the dialogue is represented can be found in [Poesio, 1994], for CRT, [Traum, 1994], for EBTL, and [Ferguson, 1995], for the plan graphs.

	7.1-7.2	U: So there's an engine at Avon.
(6)	7.3	U: Right?
	8	S : Right.

The Parser produces the underspecified representation shown in (7) for `7.1-7.2`. It can be thought of as a parse tree whose leaves have been replaced by the semantics of lexical items. Non-leaf nodes are annotated with a rule name and the corresponding semantic rule. The initial numerals 1, 2, ... indicate the first subconstituent, second subconstituent, and so on. As explained in section 4.1, the rule names S_N2THERE+V2, V2THERE-SING_VBE+N2+P2, *etc.* reflect the left-hand side and right-hand side categories occurring in the syntactic part of the rules. The semantic rules specify the formation of EL expressions, using Lisp expressions of the form (<key> <item>+), where <key> indicates the type of semantic object represented by the expression. In particular, the key :F indicates a functional application of the first argument to the rest, the key :P indicates a predicate application of the the first argument to the rest, the key :I indicates a formula in infix form (with the "subject" argument first, followed by the predicate, followed by any additional arguments), the key :L indicates a lambda-expression, the key :O indicates an unscoped operator (here, a present tense operator), and the key :Q indicates an unscoped quantifier. Scoped quantifiers are used directly as keys, *e.g.*, :E indicates existential quantification.

```
(7)     ((1 UTT 1
          (1 S-TELL (:F :DECL 1)
           (1 S_CONJCOORD+S (:I :UTT-IMP 1 2) (1 SO1 :SO-COORD)
            (2 S_N2THERE+V2 2 (1 THERE1 :EXISTS)
             (2 V2THERE-SING_VBE+N2+P2 (:E :Y (:I :Y 2) (:I :Y (:F 1 3)))
              (1 ^S2 (:L :X (:O :PRES :X)))
              (2 PRED_DETAN+N1 2 (1 AN1 :E)
               (2 ENGINE1.1 :ENGINE))
              (3 P2_PPLACE+N2 (:P 1 2) (1 AT1 :AT-LOC)
               (2 AVON1 :AVON)))))
           (2 /PERIOD1 NIL))))
```

Thus the top-level node is an UTT (utterance) node whose LF in EL is just that of its first and only constituent, a punctuated declarative sentence (S-TELL). The semantic rule for this S-TELL applies the :DECL operator to the EL formula for the unpunctuated sentence. As previously explained, this :DECL operator signals a particular illocutionary act (a 'telling'). The unpunctuated sentence, formed by rule S_CONJCOORD+S, has as its first constituent a conjunctive coordinator, *so*, and the second constituent is a sentence. The EL semantic rule specifies formation of an infix formula which relates an implicitly referenced utterance, :UTT-IMP, to the content of the current sentence, using the meaning of the coordinator as the relation. Without the *so*, we have a sentence labeled with S_N2THERE+V2, *i.e.*, one consisting of an existential *there* subject and a verb phrase (V2, meaning 'V 2-bar'). This verb phrase consists of the copular verb, a noun phrase (N2) subject in verb complement position, and a locative propositional phrase (P2). The EL formula for the sentence is just that of the verb phrase – the *there* is ignored. The verb phrase in this case is interpreted as a sentence rather than a predicate, because of the "embedded" subject (here, *an engine*); the EL semantic rule specifies existential quantification of the embedded subject, and applies a predicate formed from the interpretation of the copula (here *is*, which in essence supplies a present tense operator) and the locative P2 (here, *at Avon*).

The further interpretation into CRT does not directly apply the EL semantic rules, but rather converts the Parser output into a CRT expression that represents the (surface) *conversational event*, or locutionary act, resulting from the production of that utterance. This expression is added to the DRS representing the current discourse situation. As the syntax of CRT is a superset of that of EL, this conversion is mostly straightforward; see Section 6.1 for some further discussion. The Scoping and Deindexing (SAD) module uses the information contained in the resulting DRS to do inferences that resolve contextually-dependent expressions. This results in one or more hypotheses about the nature of the conversational events that are implicit in the given surface conversational event, taking into account the discourse context. The 6-part hypothesis obtained by processing (7) is shown in (8).

```
(8)     (:SIT-DESCR (:PAR :*COA994* :CONV-THREAD)
              (:DRS (:CE993 :CE995)
                 (:EV-DESCR :CE993 (:DRS NIL (:I :HUM :SO-INIT)))
                 (:I :CE993 :AT-ABOUT :NOW12)
                 (:I :CE993 :SUBSIT (:PAR :*COA994* :CONV-THREAD))
                 (:I :CE995 :AT-ABOUT :NOW13)
                 (:I :CE995 :SUBSIT (:PAR :*COA994* :CONV-THREAD))
                 (:I :CE993 :BEFORE :CE995)
                 (:EV-DESCR :CE995
                     (:DRS NIL
                         (:I :HUM :TELL :SYS
                             (:SIT-DESCR (:PAR :*S996* :SIT)
                                (:DRS (:E991 :X989)
                                   (:EV-DESCR :E991
                                       (:DRS NIL (:I :X989 :AT-LOC :AVON)))
                                   (:I :E991 :AT-ABOUT :CE995)  (:I :E991 :SUBSIT :S*)
                                   (:I :X989 :ENGINE)))))))
        (:= (:PAR :*RES-SIT992* :SIT) :PLAN1)
        (:= (:PAR :*RES-SIT990* :SIT) :MAPS)
        (:= (:PAR :*COA994* :CONV-THREAD) :COA1)
        (:I :CE995 :AGENT :X989)  (:I :CE995 :THEME :AVON)
```

The expression in (8) states, roughly, that (:PAR :*COA994* :CONV-THREAD) (a situation, more particularly a conversational thread — see Section 6.1) has been augmented with the two conversational events :CE993 and :CE995. :CE993 corresponds to the utterance of "so" and :CE995 to the utterance of "there's an engine at Avon".

The Lisp-like syntax used in the implementation of CRT is shown in the example. A DRS with discourse markers $a \dots b$ and conditions $\Phi_1 \dots \Phi_n$ is represented in this syntax by an expression of the form (:DRS (:A ... :B) $\Phi_1 \dots \Phi_n$). The expression s:Φ asserting that Φ holds at situation s is represented by the Lisp list (:SIT-DESCR :s Φ); the expression (:EV-DESCR :s Φ) states that Φ is a *complete* characterization of s and corresponds to the expression $[\Phi^{**}\ s]$ of EL. For example, the expression (:EV-DESCR :CE993 (:DRS NIL (:I :HUM :SO-INIT))) in (5) asserts the occurrence of an event :CE993 completely characterized by the situation type (:DRS NIL (:I :HUM :SO-INIT)), i.e., as an event of the agent :HUM performing an action of type :SO-INIT.

Some of the facts listed for (:PAR :*COA994* :CONV-THREAD) are temporal ones (*e.g.*, that the two utterances occurred at respective time points :NOW12 and :NOW13, and some are about situation structure (*e.g.*, that both utterances are part of (:PAR :*COA994* :CONV-THREAD)). Most importantly, an event description is given for each of the two utterance events. The event description for :CE993, the utterance of "so", merely notes that the agent :HUM produced such an utterance, without as yet attempting to assign a detailed logical form. The event description for :CE995 says that this is an event of the agent :HUM telling the agent :SYS that a situation to be contextually determined, (:PAR :*S996* :SIT) satisfies a certain description. This description asserts that there is an eventuality :E991 of an entity :X989 being located at Avon; it further asserts temporal and situation-structure facts, and that :X989 is an engine.

All the aspects of the interpretation which were initially underspecified have been resolved. The only traces left of the initial underspecified interpretation are the *parameters*– terms which are used to represented anaphoric 'holes' in the interpretation, as seen in section 4.2. Lisp expressions of the form (:PAR :x <TYPE>) where <TYPE> is the type of the parameter are used to represent parameters for which the syntax $\dot{x}$ or $\dot{y}$ was used in Section 4.2. For example, the expression (:PAR :*COA994* :CONV-THREAD) indicates a parameter of type CONVERSATIONAL-THREAD, and the expression (:PAR :*S996* :SIT) indicates a parameter of type situation. These parameters are 'anchored' to some specific value; for example, the second-last line of the description of (:PAR :*COA994* :CONV-THREAD) equates this discourse segment with a previously established conversational thread, :COA1, so the indeterminacy has been resolved.

Both the SAD hypothesis (8) and the underspecified representation in (7) are fed to the Speech Act Interpreter, which produces a set of core speech act alternatives, shown in (9) and a list of argumentation acts, shown in (10).

```
(9)     (:SEQ (:SURF-INTERP [CE993] [ST-ACCEPT-0028])
              (:SURF-INTERP [CE995] (:OR (:EX-OR [ST-INFORM-0029] [ST-CHECK-0030] [ST-YNQ-0031])
                                         [ST-SUGGEST-0032])))
```

```
(10)    ((:THE ?PA*T-SPEECHACT ?PADM*T-ANYTHING (:PREV-CE [CE993] ?PA*T-SPEECHACT)
             (:THE ?NA*T-SPEECHACT ?NADM*T-ANYTHING (:NEXT-CE [CE993] ?NA*T-SPEECHACT)
                   (:SO ?PA*T-SPEECHACT ?NA*T-SPEECHACT))))
```

In general, there can be many interpretations for an utterance, and the manner in which they can provide an interpretation for the utterance is specified in an alternative list of the core speech act interpretations. This list relates the conversational event to its interpretations via the predicate SURF-INTERP. As there are two sequenced conversational events (*i.e.*, utterance of "so" and "there's an engine at Avon"), a SEQ operator is used to allow sequential contextual speech act pruning and update. The manner in which the alternatives can combine is specified by the operators AND, OR, and EX-OR. The interpretation of CE995 is that it is a suggestion, ST-SUGGEST-0032 and/or one (and only one) of the following: an inform, ST-INFORM-0029, a check ST-CHECK-0030, or a yes/no question, ST-YNQ-0031. As for CE993, the "so", it is interpreted as possibly being an acceptance ST-ACCEPT-0028. This is the only core speech act conjectured for "so", but note that the argumentation act in (10), also produced from CE993, says that there is a :SO relation between the previous act, and the next act, which is CE995. The speech act definitions for [ST-ACCEPT-0028], [ST-INFORM-0029], and [ST-SUGGEST-0032], expressed in EBTL, are shown in (11), (12), and (13), respectively.

The definition of [ST-ACCEPT-0028] in (11) requires that this *accept* speech act has the agent :HUM as a speaker, :SYS as a hearer, and has as content (*i.e.*, the content to be accepted) a speech act that was produced by :SYS and was addressed to :HUM and remains unaccepted so far. The [ST-INFORM-0029] definition in (12) specifies as content of the act that there is an engine at Avon. Speaker, hearer, and time roles which are identical to those of (11) have been omitted. The (13) definition is similarly abbreviated. The definitions for [ST-CHECK-0030] and [ST-YNQ-0031] are identical to that of [ST-INFORM-0029].

```
(11)    (:AND (:ROLE [ST-ACCEPT-0028] :R-SPEAKER [HUM])
              (:ROLE [ST-ACCEPT-0028] :R-HEARER [SYS])
              (:ROLE [ST-ACCEPT-0028] :R-TIME [F-TIME [CE993]])
              (:CONTENT [ST-ACCEPT-0028]
                  (:LAMBDA ?SA*T-SPEECHACT
                     (:AND (:ROLE ?SA*T-SPEECHACT :R-SPEAKER [SYS])
                           (:ROLE ?SA*T-SPEECHACT :R-HEARER [HUM])
                           (:OCCURS ?SA*T-SPEECHACT)
                           (:UNACCEPTED ?SA*T-SPEECHACT)))))
```

```
(12)    (:AND (:ROLE [ST-INFORM-0029] :R-TIME [F-TIME [CE995]])
              (:CONTENT [ST-INFORM-0029]
                  (:LF-EXISTS ?v13237*T-ENGINE [X989] NIL
                      (:AT ?v13237*T-ENGINE  [AVON] [E991])))
              (:FOCUS [ST-INFORM-0029] [X989]))
```

```
(13)    (:CONTENT [ST-SUGGEST-0032]
            (:THE ?P*T-PLAN  ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN)
                 (:LF-EXISTS ?VAR*T-ENGINE [X989] NIL
                    (:USES ?VAR*T-ENGINE (:AT ?VAR*T-ENGINE [AVON] [E991]) ?P*T-PLAN)))))
```

During speech act pruning, [ST-ACCEPT-0028] is ruled out because there is no unaccepted suggestion by the system. (Thus only the argumentation-act interpretation of "so" remains.) Also, [ST-INFORM-0029] is ruled out because the system *did* find an engine at Avon in the **Shared** belief context ([ENGINE-1]). This leads to a decision that [ST-CHECK-0030] was the informational act performed. Evaluating [ST-SUGGEST-0032] leads to the plan reasoner call in (14). Here, the dialogue manager determined that [PLAN-4719] was the current plan, the context was the focused node in the plan graph (shown in Figure 5 from the preceding planner call, as indicated by the :so argumentation act, and the background information is that the engine is useful because it is at Avon. The planner returns the EBTL formulae in (15), which indicates that this engine fills the engine role for the focused action.

```
(14)    (INCORPORATE (:USE-OBJECT [X989]) [PLAN-4719]  :INCORP-TYPE :ROLE-FILLER
             :CONTEXT #EVENT-OLD<[MOVE-CAR-7867]>  :FOCUS [X989] :BG
             (:AT [X989] [AVON] [E991]))
```

```
(15)    (:PLAN-FACT (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
        (:PLAN-SUPPORTS
            (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867])) [MOVE-CAR-7867] [PLAN-4719])
        (:PLAN-PREMISE (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867])) [PLAN-4719])
        (:AND (:EQ [X989] [F-ENGINE MOVE-CAR-7867]))
```

The forms in (15) mean, respectively, that this equality is a fact (as opposed to an event) in the plan, that it supports the move-car event (by identifying a necessary role), that it is

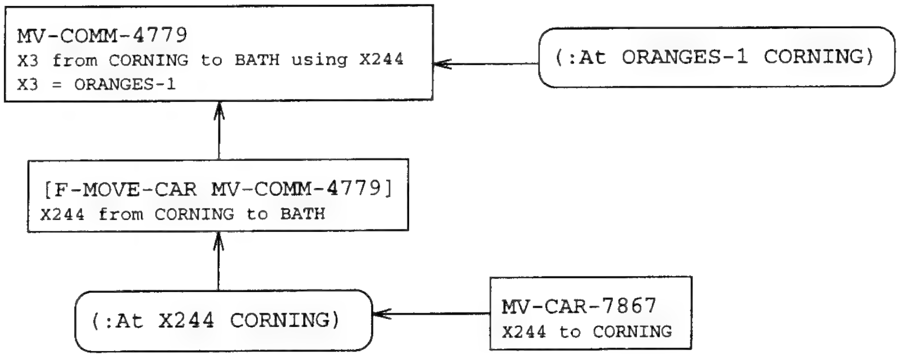


Figure 5: Plan Graph before Utterance 7.1-7-2

a premise (*i.e.*, it does not need to be achieved through further planning), and finally that the equality itself is present.

The successful incorporation lets the dialogue manager decide that [ST-SUGGEST-0032] was a valid interpretation. In addition, the dialogue manager determines that this utterance initiated a new discourse unit which would later need to be acknowledged [Traum, 1994], and that the user kept the turn.[6] The final set of conversation acts determined to be performed in performing this utterance are shown in (16), where :CSAS are the core speech act interpretations, :ARGS is the argumentation act, :GAS refers to the grounding acts, and :TTAS refers to the turn-taking acts.

(16)
```
:CSAS ([ST-CHECK-0030] [ST-SUGGEST-0032])  :ARGS (SO [CE721] [CE995])
:GAS (INIT HUM DU-4 ([ST-CHECK-0030] [ST-SUGGEST-0032]) NIL)
:TTAS (KEEP-TURN HUM [NOW13])
```

Since the user kept the turn, the system waits for the next utterance, 7.3. The parser's output for this sentence is shown in (17). This analysis views the utterance, *right?*, as an elliptic sentence, consisting just of a predicate, followed by a question mark and thus acquiring interrogative force. The top-level :QUES operator, like :DECL in the case of assertions, supplies the type of the locutionary act (an 'ask' act). A present tense operator is taken to be implicit in this type of abbreviated sentence, as can be seen in the semantic rule for the S_PRED constituent. This rule also introduces an implicit pronoun (:IT-IMP), *i.e.*, the meaning of the question is regarded as equivalent to *Is that right?*

(17)
```
(1 UTT 1
  (1 S_SELL+? (:F :QUES (:F :? 1))
   (1 S_PRED (:O :PRES (:I :IT-IMP 1))
    (1 PRED_A2 1
     (1 A2_A1 1 (1 RIGHT1 :RIGHT-CORRECT))))
   (2 /QUESTION-MARK1 NIL)))
```

[6] The user can signal a desire to keep the turn in one of two ways. Either typing an explicit signal <kt> in the input stream, or typing the next utterance before the system has finished processing the previous one.

This is converted to the CRT form shown in (18).

```
(18)    (:SIT-DESCR (:PAR :*COA1090* :CONV-THREAD)
              (:DRS (:CE1091) (:I :CE1091 :AT-ABOUT :NOW14)
                 (:I :CE1091 :SUBSIT (:PAR :*COA1090* :CONV-THREAD))
                 (:EV-DESCR :CE1091
                    (:DRS NIL
                       (:I :HUM :ASK :SYS
                           (:SIT-DESCR (:PAR :*S1092* :SIT)
                               (:DRS (:E1088)
                                  (:EV-DESCR :E1088
                                      (:DRS NIL (:I (:PAR :*IT-IMP1087* :IT-IMP)
                                                    :RIGHT-CORRECT)))
                                  (:I :E1088 :AT-ABOUT :CE1091)
                                  (:I :E1088 :SUBSIT :S*))))))))
        (:= (:PAR :*RES-SIT1089* :SIT) :PLAN1)
        (:= (:PAR :*S1092* :SIT) :COA1) (:= (:PAR :*COA1090* :CONV-THREAD) :COA1))
```

The only core speech act possibility for this utterance is the check act with content shown in (19). This act checks whether the content of some previous act is correct. In speech act evaluation, the dialogue manager decides that this refers to [ST-CHECK-0030], from utterance 7.1-7.2. Also, at the grounding level, this utterance is seen as a **repair**, replacing the previous speech act.[7] In addition, this utterance is seen as releasing the turn, and allowing the system the chance to respond.

```
(19)     (:CONTENT [ST-CHECK-0033]
          (:THE ?CE*T-INF-SA  ?CEDM*T-ANYTHING  (:PREV-ACT [CE1091] ?CE*T-INF-SA)
                (:RIGHT-CORRECT ?CE*T-INF-SA)))
```

The dialogue manager now checks its representation of the conversational state, and according to the algorithms described in [Traum and Allen, 1994; Traum, 1994], the system first decides to acknowledge the utterances by the user, and then to act on the obligation to answer the check query. This will result in first checking to see if it agrees with the mentioned fact that an engine is at Avon, and then generating an affirmative reply. The dialogue manager sends the templates shown in (20) to the generator which produces the response labeled 8.

```
(20)    ACK ([ST-CHECK-0033] [ST-SUGGEST-0032])   Reason: (GROUNDING)
        INFORM-IF  (FOCUS RIGHT (RIGHT-CORRECT [ST-CHECK-0030]))
                    Reason: (OBLIGATION (CHECK-IF (RIGHT-CORRECT [ST-CHECK-0030])))
```

The system's response is also interpreted by the system, both to update the context as well as check to make sure the intended interpretation is the most likely interpretation. The parser produces the form in (21) for this utterance. This is of course very similar to the form produced for the previous elliptic question. The only semantic difference, induced by the difference in punctuation, is the declarative force of the utterance, as encoded by the :DECL operator.

[7] Since the previous utterance also received a check interpretation, this has no significant effect, but if, for example, the inform possibility had been chosen, this utterance would change the interpretation to that of a check.

```
(21)    (1 UTT 1
          (1 S-TELL_ELL (:F :DECL 1)
           (1 S_PRED (:O :PRES (:I :IT-IMP 1))
            (1 PRED_A2 1
             (1 A2_A1 1 (1 RIGHT1 :RIGHT-CORRECT))))
           (2 /PERIOD1 NIL)))
```

The SAD module then produces the CRT form in (22).

```
(22)    (:SIT-DESCR (:PAR :*COA1143* :CONV-THREAD)
             (:DRS (:CE1144) (:I :CE1144 :AT-ABOUT :NOW15)
                (:I :CE1144 :SUBSIT (:PAR :*COA1143* :CONV-THREAD))
                (:EV-DESCR :CE1144
                    (:DRS NIL
                       (:I :SYS :TELL :HUM
                          (:SIT-DESCR (:PAR :*S1145* :SIT)
                               (:DRS (:E1141)
                                   (:EV-DESCR :E1141
                                       (:DRS NIL (:I (:PAR :*IT-IMP1140* :IT-IMP) :RIGHT-CORRECT)))
                                   (:I :E1141 :AT-ABOUT :CE1144) (:I :E1141 :SUBSIT :S*))))))))
        (:= (:PAR :*RES-SIT1142* :SIT) :PLAN1)
        (:= (:PAR :*S1145* :SIT) :COA1) (:= (:PAR :*COA1143* :CONV-THREAD) :COA1))
```

The speech act interpreter discovers two speech act possibilities for this utterance, as shown in (23), with the contents shown in (24) and (25). Both of these interpretations are validated by the dialogue manager, which as a result, transfers the plan results from the proposed to the shared plan belief context. This utterance is also seen as acknowledging the two previous utterances by the user, as well as releasing the turn.

```
(23)    (:SURF-INTERP [CE1144] (:OR [ST-INFORM-0035] [ST-ACCEPT-0034]))
```

```
(24)    (:CONTENT [ST-ACCEPT-0034]
          (:LAMBDA ?SA*T-SPEECHACT  (:AND (:ROLE ?SA*T-SPEECHACT :R-SPEAKER [HUM])
                                          (:ROLE ?SA*T-SPEECHACT :R-HEARER [SYS])
                                          (:OCCURS ?SA*T-SPEECHACT)
                                          (:UNACCEPTED ?SA*T-SPEECHACT))))
```

```
(25)    (:CONTENT [ST-INFORM-0035] (:THE ?C*T-INF-SA ?DM*T-ANYTHING (:PREV-ACT [CE1144] ?C*T-INF-SA)
                                         (:RIGHT-CORRECT ?CE*T-INF-SA)))
```

6 Relations between the KR Languages

Not choosing a single representation formalism from the very beginning involved a risk, namely, that we would end up with a number of mutually incompatible representations. In fact, this did not happen. There are strong connections between the formalisms we use, and a number of ontological assumptions are shared, so that most translations are a straightforward matter. A number of factors conspired in keeping the formalisms we use relatively close to each other. First of all, there was an unspoken agreement by all to adopt logic-based formalisms. Secondly, as work progressed, it became more and more clear that a tight interaction between the modules was needed. In this section, we describe some of those connection points, as well as some of the ontological differences springing from the different research traditions that still make it non-trivial to combine all the reasoning into a single representation language.

6.1 Translating between the Languages

The conversion from the EL-annotated parse tree produced by the parser into the initial underspecified CRT expression is mostly straightforward. For example, the parser produces for the NP "an engine" the syntactically annotated subtree shown in (26), whose last element is the semantic object to be produced.

(26) `[NP [Det an; exists] [N engine; engine]; (:Q 1 2)]`

The resulting CRT expression, shown in (27), is almost identical, but the lexical items are eliminated from nodes such as `[Det an; exists]`, and the semantic operation is eliminated from the specification of phrasal categories, since it's part of the semantics of the CRT expression.

(27) `[NP [Det λ P λ Q (∃ x : P(x) (Q(x)))] [N` **engine**`]]`

In (27), `[Det λ P λ Q (∃ x : P(x) (Q(x)))]` is the semantic translation of "an", and **engine** is the predicate associated with the noun "engine" in the lexicon, and **x** is used as the discourse marker (see Section 6.2). One aspect of the translation which involves an actual modification is where anaphoric items are involved, since EL does not have the equivalent of parameters.

The translation from EL and CRT to EBTL takes the form of an approximation, where the more complex episode or situation constructs in EL and CRT are approximated in the weaker EBTL, in terms of events and actions. EBTL does not contain general situations or episodes as first-class objects, and so only those episodes which correspond directly to events or time intervals can be translated. Much of the time, however, no information is lost, since most domain-related episodes that arise in the dialogues correspond to simple event occurrences.

More general situations that are characterized by a more complex structure of events or facts can sometimes be approximated within the belief contexts from RHET. While not strictly a part of EBTL, reasoning about hierarchically related belief contexts which contain sets of EBTL formulae is an important part of the reasoning performed by the dialogue manager. Thus, for example, discourse segments, which are represented in CRT by plural situations (called conversational threads) are represented (in part) as belief contexts containing EBTL facts and events.

In addition, as the domain predicates in EBTL are only those used in the TRAINS domain, more general event descriptions are specialized to their precise meaning in the TRAINS domain. *e.g.*, the EL action `pick-up`, as in utterance `9-13`, which has more general applicability in a range of situations is disambiguated to a `COUPLE` action within EBTL.

Translating a plan graph, represented in EBTL, to the information that the executor uses, represented in Tyro, is again an approximation. This translation removes most of the dependencies between the events expressed in a plan. That is, it finds all the actions in the plan such that the action is an `ActionIn` the plan and considers them individually. It also uses each fact that `Enables` an action as a condition to trigger the action. In addition, the executor treats the EBTL events in the plan graphs as Tyro event types. The event types are constructed by taking the EBTL description of the EBTL event, and using that description as a lambda abstraction that is reified into an Tyro event type. By doing this, the executor can use its statistical knowledge to choose its actions (messages to agents in the simulated world) to best achieve the planned event.

The loss of information in the action converstion process is important because it allows the executor to assume independence of the events in the plan, simplifying the computation of probability. Though the relations between events are certainly important for reasoning about the likely success of the whole plan, and communicating these aspects with the user, they are not essential to determining whether the events themselves will occur. On the other hand, information lost decreases the accuracy of the conditional probabilities computed. This problem is ameliorated somewhat because the executor can report failures back to the rest of the system. If some problem with a particular event in the plan occurs, the executor could communicate back to the domain plan reasoner, which would then have the full plan argument to assist in replanning.

6.2 Interface Issues

A number of knowledge-representation issues came up in the course of building the system that transcended the functioning of any one module or representation language. We briefly discuss some of them here.

Discourse Markers

Any system for handling an extended dialogue (as opposed to isolated sentences) must maintain a notion of discourse context. Thus the SAD module maintains knowledge of temporal relations between events and a structure for referentially accessible entities, and the dialogue manager and plan reasoner maintain information about mental states and plans. Some sharing of information across these distinct representations of context is essential. If, for example, the SAD module decides that some mentioned entity is the same as one that had been described before (which, perhaps the plan reasoner had decided played a particular role in a plan), it is crucial that the other modules realize this connection. This problem is addressed by introducing common discourse markers, serving as unique names across the entire system, for entities introduced by the dialogue (typically through indefinite NPs, or implicitly through lengthier dialogue segments, as in the case of various situations and plans).

These discourse markers are an integral part of CRT and EBTL, and play a crucial role in the interface between the knowledge representation languages and modules. Their importance is best seen by looking at the interface between the parser, SAD, and speech act interpreter.

Discourse markers are introduced into the unscoped logical form produced by the parser, corresponding to existentially quantified and *the*-quantified unscoped terms. This modified parser output becomes the input for both the SAD module and the speech act interpreter. The speech act interpreter then builds a skeleton of the speech act in terms of the discourse markers present in the parse tree. For the content, it turns to the CRT output of the SAD module. The merging process converts the CRT definitions for these discourse markers into EBTL, and then puts the markers and the EBTL definitions into the appropriate spots in the speech act output. In addition, these same discourse markers allow reference to objects across sentences, and in both the CRT and EBTL knowledge bases.

Definite Description

Resolving definite descriptions is a venerable problem in natural language interpretation. It is clearly necessary to represent definite description more or less directly in the underspecified logical form, so that different possibilities for the referent can be considered and chosen between, using various semantic, contextual, and inferential processes, as is done in the SAD module in TRAINS. What is less obvious is that the same kind of definite quantifier construct is also useful for the modules that reason about the discourse pragmatics and plan execution.

While explicit definite descriptions coming from utterances (*e.g., the boxcar*) could in principle be eliminated when the referent is resolved by the SAD module, it is still usefull to have a definite description operator in EBTL for representing several aspects of the contextual pragmatics in the hypotheses built by the speech act interpreter. These include the following:

- suggestions without explicit goals are taken to be implicitly suggestions with respect to some plan that is contextually relevant. The speech act interpreter, which builds hypotheses without reasoning about the context, uses the construct in (28) to signal the dialogue manager that the plan ?P, referred to in the body of the suggestion, should be the current plan in the current discourse segment.

  (28) `(:THE ?P*T-PLAN ?DM*T-ANYTHING (:CURRENT-PLAN ?P*T-PLAN)),`

- argumentation acts which make reference to component speech acts that are not part of the current utterance are augmented with definite reference information that will allow the dialogue manager to select the appropriate acts. For instance, this is done in the case of the argumentation act coming from cue word *so*, as represented in (10), in the previous section. Here, the dialogue manager can decide which are the appropriate previous and next acts which are connected by the argumentation act.

A process similar to definite reference resolution is also performed by the plan executor, when trying to monitor execution of the plan produced by the plan reasoner. The plan reasoner reifies future events in plans for the purpose of reasoning about, *e.g.*, enabling conditions, effects, *etc.* However, the plan executor, while trying to actually cause an event in the plan like [Move-Boxcar-1], views this event as an event type with some qualifications. Moreover, in monitoring the plan, the executor receives from the agents in the world a huge list of reports of things that happened during a particular stretch of time, and it needs to decide if these things mean that the intended [Move-Boxcar-1] has actually occurred. If two events of this type have occurred and they differ only on irrelevant details, it can be difficult to determine which of them was [Move-Boxcar-1], much as in the case of multiple referents fitting a definite description. So, although no definite description operator is involved here, the process is very similar to definite reference resolution, in which presuppositions of identifiability and uniqueness (within a given context) guide the selection of the appropriate object.

A related problem is how to interpret existential *there* – see for instance utterance 3.7, *There are oranges at Corning.* Originally the EL interpretation treated the "lowered subject", *oranges*, as having the same denotation as it would as an ordinary bare-plural subject, namely, the kind, (K (*plur orange*)). But the fact that the resultant LF's provided an existential entailment only indirectly, and that sentences such as *Pencils are in the top drawer*, and *There are pencils in the top drawer* differ in meaning persuaded us to treat the lowered subjects as *predicates*; we then let *there is/are* introduce explicit existential quantifiers over the domains of these predicates.

As such, there is no need for the SAD module to try to identify these oranges – it merely adds a discourse marker so that future reference to this object can be recognized. At the pragmatic level, however, this utterance, in the context of the problem solving dialogue, is interpreted as (among other things) a *suggestion* to use the oranges in the plan. Thus, seen from the point of view of the dialogue manager, this is very like a definite reference, which is resolved by querying the database to see which oranges are there (in this case, O1). When the suggestion is incorporated into the plan, O1 is used, and equated to the discourse marker so that future references will be recognized.

Indirect Action

Although the TRAINS dialogues contain locutions such as *we should move the engine*, in which the conversants refer to themselves as the agents of domain actions, as mentioned above, in the TRAINS domain, neither the user nor the system can effect changes on the world directly. Such locutions are part of a general phenomenon of indirect agency, in which one agent (*e.g.*, the system) can perform an action by having another agent (*e.g.*, the train engineer in the simulated world) do something. This brings up the point of where and how the translation from the system and user, mentioned explicitly in the dialogue, to the actual agents in the world should take place. Given the focus on domain independent language translation, it is clear that this translation cannot occur in the initial logical form – one could very well imagine the same conversations occurring in a slightly altered domain where system and user had direct control over the domain actions (*e.g.*, by moving trains represented in a GUI showing the map). Even at the conversation level, it is necessary to represent the original agents, so that the dialogue manager can track the obligations to perform particular actions. While the domain planner has no need to represent the system and user, it also is not hindered by leaving them in the agent roles, assuming such actions actually can be performed in the world.

It is up to the plan executor to finally resolve this issue by recognizing that the actions specified in the plans correspond to requests to the simulated agents. To do this, it must examine the event specified in the plan and reason about which agents in the domain should be involved. For example, if the event involves moving a railroad car, the executor knows that the only agents that can really make a car move are train engineers. It also knows that it communicates with engineers through their engines, so it looks for an engine mentioned in the event.

In fact, this problem is more general than just a case of "sloppy" or metaphoric locution. This kind of indirect agency is also sometimes expressed directly, as in the following utterance from another TRAINS dialogue, "We'll have E3 pick it up". For this reason, a *bring-about* event was incorporated in both EL and EBTL. In fact, even the verb *ship*, in our first sentence is most properly translated as bringing about a move-commodity event.

This raises a related issue in sentences like 1, from Figure 2 where there is a temporal qualification of "by 8 a.m." on the ship action. The question arises as to whether this refers to the *ship* action itself or the underlying move-commodity action. While, in general, the locution is ambiguous, with, perhaps a preference for the former (*e.g.*, "We have to ship this package by 5pm, before UPS closes."), in examining the TRAINS corpus, all temporal qualifications of *ship* refered to the conclusion of the embedded move-commodity event.

6.3 Conflicting Ontologies

While there is need for more interaction and information sharing between the modules of the TRAINS system than was present in the TRAINS-93 implementation, (*e.g.*, the SAD module needs to know the results of Speech Act interpretation, as these affect discourse segmentation and therefore the availability of anaphoric references), it is by no means clear that in the future all modules of the system will be able to access a unified database of information. In order for that to happen, it is not enough to adopt a single representation formalism; it is also necessary that all modules adopt the same ontological perspective, and it is by no means clear that that's possible, or even desirable. We mention below a couple of issues in which the most natural view of some important phenomenon is different for modules concentrating on different aspects of the planning dialogue task.

Common Ground

Most work on reference resolution makes the assumption that felicitous reference requires mutual knowledge [Clark and Marshall, 1981], and that, further, everything that is uttered in a shared situation immediately becomes a part of the common ground and, hence, is available for reference. This assumption was carried over into CRT , so that, *e.g.*, a speaker can refer to an object mentioned in the immediately preceding utterance.

As shown in [Clark and Wilkes-Gibbs, 1986; Clark and Schaefer, 1989; Traum and Hinkelman, 1992], this assumption of immediate addition to common ground is an idealization: utterances by one party must be recognized and acknowledged by the other before becoming part of the (assumed) common ground. It is important for a system engaging in extended dialogues (rather than just iterative question answering exchanges) to have an awareness of the grounding process, so that acknowledgments and repairs can be both recognized and produced when appropriate. Grounding is tracked by the dialogue manager (the current state accessible through EBTL predicates) and used both for recognition and production of utterances.

Because of the different interpretations of the common ground and the importance of the concept for various tasks (definite reference resolution on the one hand, and producing and interpreting acknowledgments and repairs on the other), there are two parallel representations of much of the discourse context. Thus, the SAD module represents discourse segments in CRT, using situations, while the dialogue manager represents them in EBTL using RHET's belief context mechanism.

What are *Plans?*

Plans are obviously one of the most important entities to be represented in a system designed to engage in planning dialogues. However, the most useful perspective on the plan varies from module to module, resulting in different ontologies of plans in the different representation languages. This distinction goes far beyond the by now familiar distinction between plans as recipes and plans as mental state [Bratman, 1987]. The different views of plans and their rationales are summarized below.

Parser/Semantic Interpreter: From a natural language perspective, the semantic category of plans should be whatever category is assigned to them by locutions that directly and uncontroversially specify them. The clearest examples of such locutions are those like "Our current plan is to fill tankers T3 and T4 with beer", "The plan is for engine E3 to go to Dansville and pick up the boxcar there", or "Our plan of going to Bath and picking up both boxcars won't work", or "The plan is that we'll bring a boxcar of oranges to Elmira, take them to the factory, and produce OJ". Arguably, the infinitive in the first sentence (*to fill . . . with beer*), the *for*-clause in the second (*for engine E3 to go . . . the boxcar there*), the gerund in the third (*going to Bath . . . both boxcars*), and the *that*-clause in the fourth (*that we'll bring . . . and produce OJ*) are all *nominals*; *i.e.*, syntactically, they occupy noun phrase positions and semantically they denote individuals of some sort. But it can be further argued that nominalized infinitives, nominalized *for*-clauses and gerunds denote *kinds/types of actions* (or kinds/types of situations/events), and that *that*-clauses denote *propositions* (e.g., see [Chierchia, 1985; Hwang, 1992]). The difference between viewing plans as situation types and viewing them as propositions is minor in Episodic Logic: a proposition determines a situation type, namely the type of situation in which the proposition holds; and conversely, a situation type determines a proposition, namely the proposition which is true precisely in situations of that type. Thus the most natural assumption to make about linguistically specified plans is that they are types of situations, or (more or less equivalently) propositions.

SAD: Another important aspect of plans is that, just as with the visual situation (or map — Figure 1 in the TRAINS domain), and the conversational threads that are part of the discourse situation, they can serve as a resource situation for referring expressions. There are some objects (*e.g.*, orange juice to be created later) or qualifications (*e.g.*, "the boxcar at Corning", when there will be such a boxcar only after performance of some planned actions) which are present only in plans. In order to treat coreference uniformly — whether it is between expressions in descriptions of actual events, or between expressions in descriptions of planned events, or (as often happens) between expressions one of which occurs within a description of a given situation, while the other occurs within a description of a planned situation — the SAD module treats a plan as a situation, just like the others — in this case containing (potential) future objects and events and the way the map will be as a result of the occurrence of the events in the plan. While some of the ways of talking about a plan, such as those given above or the ability to perform temporal modificiation (*e.g.*, doing the plan at noon rather than morning, or doing it twice in a row), might indicate that a plan should be a *situation type* rather than a situation, the *situation* view has other practical considerations. When people plan they often seem to (sometimes tentatively) commit as they go, so that the planned steps are presented as if they were simply descriptions of future events.

Dialogue Manager: From a dialogue management perspective, the most salient aspect of plans is that they are constructed and their content negotiated through the dialogue process. It is not important for dialogue management purposes to focus on the details or correctness of plans, but rather that conversants can have different views of a plan, that the composition of a plan can change through time as the conversation progresses (both through the addition of further details, as well as modification of existing components), and that, in spite of these differences, in some sense the conversants are talking about the *same* plan. For these reasons, the dialogue manager views plans as abstract objects that, like physical objects, may have different properties at different times oraccording to different agents. Plans are treated as individuals in the logic, with properties expressed by plan predicates described in Section 4.3. A plan is represented as an EBTL term, along with a collection of predicates about the term, with different sets in different belief contexts.

Plan Reasoner: Embedding of the plan reasoner within a *mixed-initiative* planning setting (where each participant may contribute parts, and must recognize and agree to the contributions of others) has implications for how plans themselves are viewed. Since the plan reasoner must reason about the intended relations between mentioned actions and states within a plan (whether or not these relations would actually hold), as well as *evaluating* plans, and producing augmentations to be proposed to the user, and all of these processes are defeasable, the plan reaoner must be able to represent incorrect plans, as well as the fact (and reasons why) the plans are incorrect. Also, plans need only be specified as far as necessary to convince the user and system of their feasibility. These requirements make it natural to view a plan as an *argument* that a given goal will be achieved, given certain actions and assumptions. The system and user can incrementally refine their plans just as arguments are refined in response to criticisms or perceived deficiencies. The argument structure of a plan is represented as a DAG, as described in Section 4.3.

Executor: The plan executor's task of sending messages to agents in the world requires a focus on a yet another different aspect of plans. From this point of view, the plan is a partial specification of behavior. Unlike the language modules or even the plan reasoning module, the executor cannot treat the plan as a single situation. Instead, it must extract from the plan the points at which the ongoing behavior of the world can be modified to meet the plan's specifications. To make the plan occur, the executor must first decompose the plan into a set of events whose occurrence it can affect. Next, it must reason about how to make these events happen. Usually, the events occur when the executor sends requests to agents that can act to cause the events directly. Finally, it must note the expected effects so it can monitor the occurrence of these events. Currently, because the executor does not report the failure of plans back to the rest of the system, there is no reason to record that the events it is monitoring are part of a plan; monitoring only gathers information that will be used to reason about subsequent events of the same type. If, however, the executor were to report the failure of the plan, it would have to record the plan that gave rise to the events with the events it is monitoring. As it stands now, however, a plan is a set of temporally ordered desirable events, which can be translated into a sequence of requests for an appropriate agent to act in such a way that to the specified event will occur in the simulated world

While none of these are completely incompatible, unification into a single representation language must be performed carefully, in such a way that each module can perform its own functions with as great a facility as in the current system. Given the divergent perspectives, it is more straightforward to insure consistency within separate languages than within a single one.

7 Assessing the KR in the System

Concerning syntax, we previously noted our use of a GPSG-like grammar in the parsing / LF-computation module. Of course, phrase structure grammars are thought to fall slightly short of the expressive power needed in general for the syntactic characterization of human languages. However, this appears not to be a serious impediment to building broad-coverage grammars of English, and the simplicity of chart parsing for phrase structure grammars make their use almost irresistible for large-scale systems. Some more serious problems for syntax and parsing, which we deliberately set aside for TRAINS-93, are the fragmentation of sentence structure through turn-taking; false starts and other disfluencies; the role of prosody; and more generally, the problems raised by interactive *spoken* language, such as how to deal with the inevitable word recognition errors.

Among implemented NL systems that use a formal meaning representation, the great majority rely on description logics or frame-like representations. By and large, these representations capture only a subset of the expressive power of standard FOL, and in terms of their inference capabilities are aimed primarily at subsumption checking [Brachman and Levesque, 1984; Nebel and Smolka, 1991; Schmolze and Israel, 1983; Hayes, 1979].[8]

[8] In lexically oriented work such as [Dorr, 1993], Jackendoff's Lexical Conceptual Structures are gaining

Our KR formalisms, especially the EL and CRT representations for language, were seen to be considerably more expressive than FOL. One might ask whether in retrospect the purposes of our project might have been equally well served by a weaker representation. We think this would have made our language module less general, while at the same time making it harder to implement and extend. As ever, the devil here lies in the details, so let us briefly consider utterance, `5.1`, from our dialogue.

The only way to provide an approximate representation of this in standard FOL (without quantifying over possible worlds) is to leave out a great deal. In particular, as shown in the example, the *so* appears to establish a relevance connection or a "following-from" connection between a previous proposal and the present one, but FOL does not allow us to talk about this sort of sentential or propositional connection. Further, *need* is patently intensional. So, ignoring both, we might claim that in essence, the utterance posits a future *move*-event whose instrument is some engine and whose theme is the boxcar. This readily admits an FOL-representation.

Now the neglect of a weak cue word like *so* is probably harmless in many cases; but such a strategy does not generalize well. For instance, an initial *but* can be used to call into question a previous suggestion, as in

"First let's get rid of the tanker cars."

"But we need a tanker car to ship the orange juice."

So neglect of cue words is bound to impair discourse understanding, and we would probably end up treating them as a separate phenomenon with a separate representation, losing in simplicity and generality. Similarly the neglect of intensional verbs like *need* would come back to haunt us. If we replace it by simple future, we may misunderstand a goal statement as a prediction; if we leave it in place but treat it as non-intensional, we will incorrectly conclude that there is a *particular* engine that is needed to move the boxcar, and may well ask, "Which one is it?"

The TRAINS dialogues are full of such locutions defying simple FOL representations. As a few more examples (besides those mentioned in section 3.1) drawn from various transcripts, consider "That's reliable information" (the subject of the predication is a proposition); "... our next objective here which is to deliver a boxcar of bananas to Corning" (the "objective" is equated with an action type); "Problem number two looks difficult" ("looks difficult" is not equivalent to "is difficult", and creating a new predicates like *looks-difficult* would severly inflate the lexicon and KB); "What is the best way for me to accomplish my task?" (the referent of "the best way ..." is an action type or plan); "That's a little beyond my abilities" (the subject is an action type, and "a little" is a predicate modifier not reducible to a conjunct); "Our current plan is to fill tankers T3 and T4 with beer" (the NP subject and the infinitive complement refer to the same thing, a plan).

If we truly want to understand such locutions, then reliance on an expressively weak representation would make it much more difficult to find any sort of approximate LF, and we would in addition have to fill in the gaps left by these approximations in various *ad hoc* ways. We would lose doubly, in both simplicity and generality.

some currency. However, inference based on LCS's again tends to be restricted to subsumption, implemented by translating into a description logic such as CLASSIC.

Contrary to a widespread myth a rich syntax is no obstacle to effective inference. For instance, although only a very limited set of EL inference capabilities has been used in TRAINS-93, EL has been separately implemented in the EPILOG system [Schaeffer et al., 1993]. EPILOG is a powerful knowledge management and inference system allowing data-driven inference, goal-driven inference, and featuring integration with about a dozen specialist modules for accelerating temporal, taxonomic, partonomic, set-theoretic, numeric, and other special types of inference.

In general, in building a complex system containing multiple knowledge representations, it is better to err on the side of "overly rich" representations than on the other side. The reason is that when information is shipped from one module to another, it is far easier to discard superfluous information than to make up for missing information, omitted for lack of sufficient expressive power. This has been strongly confirmed by our experience, where the rapid evolution of each module caused frequent changes in its information needs, and hence also in the information extracted from its "supplier" modules. The unusual expressiveness of our representations, particularly at points nearer to the input end, made these adaptive changes relatively easy."

8 Current and Future Directions

While the theories behind many of the individual modules are continuing research projects, there are also several efforts in progress or under discussion which follow up on the knowledge representation issues described in this paper. Some of these are described briefly below.

Unified Conversational Context Representation Given the experience of designing the individual modules and the interfaces, we are now in a position to meet head on the ontological differences springing from the different research traditions on discourse context representation. In [Poesio and Traum, 1995] we present the first steps, which alters CRT to also allow representation of the aspects of conversational context represented in EBTL in the TRAINS-93 system.

Robust Spoken and Multi-modal Interaction The TRAINS-95 system [Allen et al., 1995a; Ferguson et al., 1996], the most recent effort in the TRAINS project as of this writing, represents a somewhat separate effort to develop a relatively simple but extremely robust multi-modal system, based on our experiences with TRAINS. This system understands both spoken and typed input, as well as various mouse operations such as selecting from a map or menu. It generates responses using both spoken language and graphical displays.

The task for TRAINS-95 is currently a simple route planning scenario designed to encourage interaction between the human and the system, thereby demonstrating the effectiveness of dialogue-based interaction. The goal is to allow an untrained user to sit down and solve a randomly-generated problem. This puts the emphasis on robustness, since we have to deal with such phenomena as incomplete and ungrammatical utterances, speech recognition errors, multi-modal input and output, and so on. We have also concentrated on more clearly delineating the modules of the system as independent knowledge sources, reasoning agents, and display engines, allowing us to plug in more sophisticated components easily. We are currently scaling up the deliberately naive route planning domain reasoner to a more interesting and challenging transportation planning scenario.

Knowledge Acquisition by Simulation We have built a general purpose simulator for transportation domains such as TRAINS [Martin and Mitchell, 1995]. The simulator provides a graphical interface that allows us to quickly set up scenarios in the TRAINS world. It also animates the simulation to help us debug the plans the system generates. We are currently exploring techniques for incorporating simulation as a source of knowledge that the system can use to help the user develop effective plans. While evaluating the effects of a plan may be intractible in general, due to too many possible contingencies and inadequate prior information, simulation based on probablities can reveal likely problems and expectations for success of particular options.

Other We expect to eventually build a hybrid TRAINS system, containing modules similar to those of earlier prototypes, but also incorporating the speech processing modules of TRAINS-95, as well as its heuristic mechanisms for mapping very fragmentary parses into plausible speech act hypotheses. We also expect to use a mechanism for automatically detecting and correcting speech repairs, along the lines of that presented in [Heeman and Allen, 1994a].

Acknowledgments

The TRAINS project has been funded in part by ONR/ARPA grant N00014-92-J-1512, U.S. Air Force–Rome Laboratory research contract F30602-91-C-0010, and NSF grant IRI-90-13160. We would like to thank the other participants in the TRAINS project efforts who were not directly involved in the knowledge representation and reasoning components of the TRAINS-93 system, especially Brad Miller, who provided continuing support for RHET, used for EBTL reasoning, Tsuneaki Kato, who implemented the probabilistic parser used in TRAINS-93, and Graham Katz, who first pointed out the problems with the temporal constraints of *ship* described in Section 6.2.

References

[Allen, 1984] Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

[Allen and Ferguson, 1994] Allen, J. F. and Ferguson, G. (1994). Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5).

[Allen et al., 1995a] Allen, J. F., Ferguson, G., Miller, B., and Ringger, E. (1995a). Spoken dialogue and interactive planning. In *Proc. of the ARPA Spoken Language Technology Workshop.* Also available via the WWW as `http://www.cs.rochester.edu/research/trains/trains95/`.

[Allen et al., 1989] Allen, J. F., Guez, S., Hoebel, L., Hinkelman, E., Jackson, K., Kyburg, A., and Traum, D. (1989). The discourse system project. Technical Report 317, Department of Computer Science, University of Rochester.

[Allen and Miller, 1991] Allen, J. F. and Miller, B. W. (1991). The RHET system: A sequence of self-guided tutorials. Technical Report 325, Department of Computer Science, University of Rochester.

[Allen et al., 1995b] Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., Hwang, C. H., Kato, T., Light, M., Martin, N., Miller, B., Poesio, M., and Traum, D. R. (1995b). The TRAINS project: a case study in building a conversational planning agent. *Journal of Experimental and Theoretical Artificial Intelligence*, 7:7–48.

[Alshawi and Crouch, 1992] Alshawi, H. and Crouch, R. (1992). Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 32–39, University of Delaware.

[Barwise and Perry, 1983] Barwise, J. and Perry, J. (1983). *Situations and Attitudes.* MIT Press, Cambridge, MA.

[Bickel and Doksum, 1977] Bickel, P. J. and Doksum, K. A. (1977). *Mathematical Statistics: Basic Ideas and Selected Topics.* Holden-Day, Inc., Oakland, CA.

[Brachman and Levesque, 1984] Brachman, R. J. and Levesque, H. J. (1984). The tractability of subsumption in frame-based description languages. In *Proceedings of the fourth National Conference of the American Association for Artificial Intelligence (AAAI-84)*, pages 34–37.

[Bratman, 1987] Bratman, M. E. (1987). *Intention, Plans, and Practical Reason.* Harvard University Press.

[Brewka, 1991] Brewka, G. (1991). *Nonmonotonic Reasoning: Logical Foundations of Commonsense.* Cambridge University Press, Cambridge.

[Chierchia, 1985] Chierchia, G. (1985). Formal semantics and the grammar of predication. *Linguistic Inquiry*, 16:417–443.

[Clark, 1992] Clark, H. H. (1992). *Arenas of Language Use.* University of Chicago Press.

[Clark and Marshall, 1981] Clark, H. H. and Marshall, C. R. (1981). Definite reference and mutual knowledge. In Joshi, A. K., Webber, B. L., and Sag, I. A., editors, *Elements of Discourse Understanding.* Cambridge University Press. Also appears as Chapter 1 in [Clark, 1992].

[Clark and Schaefer, 1989] Clark, H. H. and Schaefer, E. F. (1989). Contributing to discourse. *Cognitive Science*, 13:259–294. Also appears as Chapter 5 in [Clark, 1992].

[Clark and Wilkes-Gibbs, 1986] Clark, H. H. and Wilkes-Gibbs, D. (1986). Referring as a collaborative process. *Cognition*, 22:1–39. Also appears as Chapter 4 in [Clark, 1992].

[Davidson, 1967] Davidson, D. (1967). The logical form of action sentences. In Rescher, N., editor, *The Logic of Decision and Action.* University of Pittsburgh Press, Pittsburgh, PA.

[Dorr, 1993] Dorr, B. J. (1993). Interlingual machine translation: A parameterized approach. *Artificial Intelligence*, 63(1,2).

[Ferguson, 1995] Ferguson, G. (1995). *Knowledge Representation and Reasoning for Mixed-Initiative Planning.* PhD thesis, University of Rochester. Also available as TR 562, Department of Computer Science, University of Rochester.

[Ferguson et al., 1996] Ferguson, G., Allen, J. F., and Miller, B. (forthcoming, 1996). TRAINS-95: Towards a mixed-initiative planning assistant. In *Proceedings of the Third International Conference on AI Planning Systems (AIPS-96)*, Edinburgh, Scotland.

[Gazdar et al., 1985] Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar.* Blackwell, Oxford.

[Goldman, 1970] Goldman, A. I. (1970). *A Theory of Human Action.* Prentice Hall Inc.

[Groenendijk and Stokhof, 1991] Groenendijk, J. and Stokhof, M. (1991). Dynamic Predicate Logic. *Linguistics and Philosophy*, 14:39–100.

[Gross et al., 1993] Gross, D., Allen, J., and Traum, D. (1993). The TRAINS 91 dialogues. TRAINS Technical Note 92-1, Department of Computer Science, University of Rochester.

[Grosz and Sidner, 1986] Grosz, B. J. and Sidner, C. L. (1986). Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.

[Hayes, 1979] Hayes, P. (1979). The logic of frames. In Metzing, D., editor, *Frame Conceptions and Text Understanding.* de Gruyter.

[Heeman and Allen, 1994a] Heeman, P. and Allen, J. (1994a). Detecting and correcting speech repairs. In *Proceedings of the 32^{th} Annual Meeting of the Association for Computational Linguistics*, pages 295–302, Las Cruces, New Mexico.

[Heeman and Allen, 1994b] Heeman, P. A. and Allen, J. (1994b). The TRAINS 93 dialogues. TRAINS Technical Note 94-2, Department of Computer Science, University of Rochester.

[Hwang, 1992] Hwang, C. H. (1992). *A Logical Approach to Narrative Understanding.* PhD thesis, University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada.

[Hwang and Schubert, 1993a] Hwang, C. H. and Schubert, L. K. (1993a). Episodic Logic: A comprehensive, natural representation for language understanding. *Minds and Machines*, 3:381–419.

[Hwang and Schubert, 1993b] Hwang, C. H. and Schubert, L. K. (1993b). Episodic Logic: A situational logic for natural language processing. In *Situation Theory and its Applications, V. 3*, pages 303–338, CSLI, Stanford, CA.

[Kamp, 1981] Kamp, H. (1981). A theory of truth and semantic representation. In Groenendijk, J., Janssen, T., and Stokhof, M., editors, *Formal Methods in the Study of Language.* Mathematical Centre, Amsterdam.

[Kamp and Reyle, 1993] Kamp, H. and Reyle, U. (1993). *From Discourse to Logic.* D. Reidel, Dordrecht.

[Koomen, 1990] Koomen, J. A. (1990). The timelogic temporal reasoning system. Technical Report 231, Department of Computer Science, University of Rochester.

[Kyburg, 1991] Kyburg, Jr., H. E. (1991). Evidential probabilities. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI '91)*, pages 103–110.

[Lehman et al., 1991] Lehman, J. F., Lewis, R. L., and Newell, A. (1991). Natural language comprehension in Soar: Spring 1991. Technical Report CMU-CS-91-117, School of Computer Science, Carnegie Mellon University.

[Mann, 1988] Mann, W. C. (1988). Dialogue games: Conventions of human interaction. *Argumentation*, 2:511–532.

[Mann and Thompson, 1987] Mann, W. C. and Thompson, S. A. (1987). Rhetorical structure theory: A theory of text organization. Technical Report ISI/RS-87-190, USC, Information Sciences Institute.

[Martin, 1993] Martin, N. G. (1993). *Using Statistical Inference to Plan Under Uncertainty.* PhD thesis, Department of Computer Science, University of Rochester.

[Martin and Mitchell, 1995] Martin, N. G. and Mitchell, G. J. (1995). TRAINS world simulator: User's manual. TRAINS Technical Note 95-2, Department of Computer Science, University of Rochester.

[Miller, 1990] Miller, B. W. (1990). The RHET plan recognition system. Technical Report 298, Department of Computer Science, University of Rochester.

[Montague, 1973] Montague, R. (1973). The proper treatment of quantification in english. In Hintikka, K. *et al.*, editor, *Approaches to Natural Language*, pages 221–242. D. Reidel, Dordrecht.

[Nebel and Smolka, 1991] Nebel, B. and Smolka, G. (1991). Attributive description formalisms ... and the rest of the world. In Herzog, O. and Rollinger, C., editors, *Text Understanding in LILOG.* Springer-Verlag.

[Poesio, 1991] Poesio, M. (1991). Relational semantics and scope ambiguity. In Barwise, J., Gawron, J. M., Plotkin, G., and Tutiya, S., editors, *Situation Semantics and its Applications, vol.2*, chapter 20, pages 469–497. CSLI, Stanford, CA.

[Poesio, 1993] Poesio, M. (1993). A situation-theoretic formalization of definite description interpretation in plan elaboration dialogues. In Aczel, P., Israel, D., Katagiri, Y., and Peters, S., editors, *Situations Theory and its Applications, vol.3*, chapter 12, pages 343–378. CSLI, Stanford.

[Poesio, 1994] Poesio, M. (1994). *Discourse Interpretation and the Scope of Operators.* PhD thesis, University of Rochester. Also available as TR 518, Department of Computer Science, University of Rochester.

[Poesio, 1995] Poesio, M. (1995). Semantic ambiguity and perceived ambiguity. In van Deemter, K. and Peters, S., editors, *Semantic Ambiguity and Underspecification.* CSLI, Stanford, CA.

[Poesio and Traum, 1995] Poesio, M. and Traum, D. R. (1995). A multi-purpose model of conversational context. In *IJCAI-95 Workshop on Context in Natural Language Processing.*

[Rambow, 1993] Rambow, O., editor (1993). *Proceedings ACL SIG Workshop on Intentionality and Structure in Discourse Relations.* Association for Computational Linguistics.

[Reiter, 1980] Reiter, R. (1980). A logic for default reasoning. *Artifical Intelligence*, 13(1,2):81–132.

[Reyle, 1993] Reyle, U. (1993). Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics*, 3.

[Ritchie et al., 1992] Ritchie, G., Russell, G., Black, A., and Pulman, S. (1992). *Computational Morphology: Practical Mechanisms for the English Lexicon.* MIT Press.

[Schaeffer et al., 1993] Schaeffer, S. A., Hwang, C. H., de Haan, J., and Schubert, L. K. (1993). EPILOG: The computational system for Episodic Logic (user's guide). Technical report, Department of Computational Science, University of Alberta, Edmonton, Alberta.

[Schegloff and Sacks, 1973] Schegloff, E. A. and Sacks, H. (1973). Opening up closings. *Semiotica*, 7:289–327.

[Schmolze and Israel, 1983] Schmolze, J. G. and Israel, D. J. (1983). KL-ONE: Semantics and classification. In *Research in Knowledge Representation and Natural Language Understanding*, pages 27–39. Bolt, Beranek, and Newman Inc. BBN Technical Report No. 5421.

[Sinclair and Coulthard, 1975] Sinclair, J. M. and Coulthard, R. M. (1975). *Towards an analysis of Discourse: The English used by teachers and pupils.* Oxford University Press.

[Traum, 1994] Traum, D. R. (1994). *A Computational Theory of Grounding in Natural Language Conversation.* PhD thesis, Department of Computer Science, University of Rochester. Also available as TR 545, Department of Computer Science, University of Rochester.

[Traum and Allen, 1994] Traum, D. R. and Allen, J. F. (1994). Discourse obligations in dialogue processing. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

[Traum and Hinkelman, 1992] Traum, D. R. and Hinkelman, E. A. (1992). Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599. Special Issue on Non-literal language.

[Walker and Whittaker, 1990] Walker, M. A. and Whittaker, S. (1990). Mixed initiative in dialogue: An investigation into discourse segmentation. In *Proceedings ACL-90*, pages 70–78.